Savitribai Phule Pune University,Pune

F. Y. B. Sc.

(Computer Science)

Laboratory Course I

Work Book

Semester-II

Name ____

College Name - K.T.H.M.College,Nashik.

Roll No. ______________________ Division _____________________

Academic Year - 2019-20.

Introduction

1. About the work book

This workbook is intended to be used by F.Y.B.Sc (Computer Science) students for the Computer Science laboratory courses in their curriculum. In Computer Science, hands-on laboratory experience is critical to the understanding of theoretical concepts studied in the theory courses. This workbook provides the requisite background material as well as numerous computing problems covering all difficulty levels.

The objectives of this book are

1) Defining clearly the scope of the course
2) Bringing uniformity in the way the course is conducted across different colleges
3) Continuous assessment of the course
4) Bring in variation and variety in the experiments carried out by different students in a batch
5) Providing ready reference for students while working in the lab
6) Catering to the need of slow paced as well as fast paced learners

2. How to use this workbook

This workbook is mandatory for the completion of the laboratory course. It is a measure of the performance of the student in the laboratory for the entire duration of the course.

2.1 Instructions to the students

Please read the following instructions carefully and follow them

1) You are expected to carry this book every time you come to the lab for computer science practicals

2) You should prepare yourself before hand for the Exercise by reading the material mentioned.

3)You will be assessed for each exercise on a scale of 5

i) Not done	0
ii) Incomplete	1
iii) Late Complete	2
iv) Needs improvement	3
v) Complete	4
vi) Well Done	5

2.3. Instructions to the Lab administrator

You have to ensure appropriate hardware and software is made available to each student.
The operating system and software requirements on server side and also client side are as given below
Operating Environment:

For 'C' Programming :
Operating system: Linux
Editor: Any linux based editor like vi, gedit etc.
Compiler : cc or gcc

For DBMS:
Operating System: Linux Operating system
DBMS: PostgreSQL
Language: SQL

3. Editors

Section A – Advanced C

Prepared By

Ms. Poonam Ponde	Nowrosjee Wadia College, Pune
Ms. Ashwini Kamble	Nowrosjee Wadia College, Pune

Updated By

Ms.Sunita N.Deore	K.T.H.M.College, Nashik
Ms.Ratna S.Chaudhari	K.T.H.M.College, Nashik
Ms.Asmita L.Taskar	K.T.H.M.College, Nashik
Ms.Priyanka T.Sawale	K.T.H.M.College, Nashik

Assignment Completion Sheet

Section A : Advanced 'C' Programming		
S.No	**Assignment Name**	**Remark**
1	Use of simple pointers	
2	Dynamic Memory Allocation	
3	String handling using standard library functions.	
4	Structure and Unions	
5	File Handling	
6	C Preprocessors	
Section B :Relational Database Management System		
S.No	**Assignment Name**	**Remark**
1	Stored Procedure	
2	Stored Function	
3	Cursors	
4	Exception Handling	
5	Triggers	

CERTIFICATE

This is to certify that, Mr./Miss.____ Roll No.________ of F.Y.B.Sc. (Computer Science) has successfully completed ______out of _____ practicals satisfactorily during the academic year 20 -20 .

Practical In-charge

Head,
Dept.Of Computer Science

Internal Examiner

External Examiner

Savitribai Phule Pune University,Pune

Lab Course I
Section A

Advanced ‘C’ Programming

Exercise 1

Start Date	/ /

To demonstrate use of pointers in C.

You should read the following topics before starting this exercise
1. What is a pointer?
2. How to declare and initialize pointers.
3. '*' and '&' operators.

A Pointer is a variable that stores the memory address of another variable

Actions involving Pointers	syntax	Example

Declaration of pointers	data_type * pointer_name	int *p1,*p2;
		float *temp1;

Initialization of pointers	pointer =&variable
	p1=&n;
		int a, *p= &a;

Pointer Arithmetic	The C language allow arithmetic operations to be performed on pointers: Increment, Decrement, Addition, Subtraction When a pointer is incremented (or decremented) by 1, it increments by sizeof(datatype). For example, an integer pointer increments by sizeof(int).	
Pointers and Functions	We can pass the address of a variable to a function. The function can accept this address in a pointer and use the pointer to access the variable's value.	
Arrays And Pointers	An array name is a pointer to the first element in the array. It holds the base address of the array. Every array expression is converted to pointer expression as follows: a[i] is same as *(a+i)	int n; *n , *(n + 0) represents 0th element n[j], *(n+ j),* (j + n) , j [n] : represent the value of the jth element of array n

1. Sample program

```
main()
{
        int a = 10, b = 20;
        void swap1( int x, int y);
        void swap2( int *ptr1, int *ptr2);

        printf(“\nBefore swapping : a=%d, b=%d”, a,b);
        swap1(a, b);
        printf(“\nAfter swapping by swap1 : a=%d, b=%d”, a,b);
        swap2(&a, &b);
        printf(“\nAfter swapping by swap2 : a=%d, b=%d”, a,b);
}

void swap1( int x, int y)
{
        int temp;
        temp = x;
        x = y;
        y = temp;
}

void swap2( int *ptr1, int *ptr2)
{
        int temp;
        temp = *ptr1;
        *ptr1 = *ptr2;
        *ptr2 = temp;
}
```

Set A . Write C programs for the following problems.

1. Write a function which takes hours, minutes and seconds as parameters and an integer s and increments the time by s seconds. Accept time and seconds in main and Display the new time in main using the above function.
2. Write a program to display the elements of an array containing n integers in the reverse order using a pointer to the array.

Signature of the instructor		Date	/ /

Set B . Write C programs for the following problems.

1. Accept n integers in array A. Pass this array and two counter variables to a function which will set the first counter to the total number of even values in the array and the other to the total number of odd values. Display these counts in main. (Hint: Pass the addresses of the counters to the function)

2. Write a function which accepts a number and three flags as parameters. If the number is even, set the first flag to 1. If the number is prime, set the second flag to 1. If the number is divisible by 3 or 7, set the third flag to 1. In main, accept an integer and use this function to check if it is even, prime and divisible by 3 or 7. (Hint : pass the addresses of flags to the function)

Signature of the instructor		Date	/ /

Set C. Write programs to solve the following problems

1. Accept date (dd, mm yy). Write a function to add no of days to the date and display the new date. Pass dd, mm and yy to the function using pointers.

Signature of the instructor		Date	/ /

Exercise 2

Start Date / /

To demonstrate Dynamic Memory Allocation

You should read the following topics before starting this exercise
4. What is a pointer?
5. How to declare and initialize pointers.
6. '*' and '&' operators.
4. Pointer to a pointer.
5. Relationship between array and pointer.
6. Pointer to array and Array of pointers.

Actions involving Pointers	syntax	Example

Pointer To Pointer	datatype **pointer_to_pointer;	int a;
		int * p;
		int **q;
		p = &a;
		q = *p ;

To allocate memory dynamically	The functions used are : malloc, calloc, realloc
	ptr = (cast-type *) malloc (byte-size) ;
	Allocates a block of contiguous bytes. If the space in heap is not sufficient to satisfy request, allocation fails, returns NULL.
	ptr1 = (cast-type *) calloc (byte-size);
	Similar to malloc, but initializes the memory block allocated to 0.
	ptr = realloc (ptr, new size);
	To increase / decrease memory size.
		int * p,*p1;
		p = (int *) malloc(10 * sizeof(int));
		p1 = (int *) calloc(10, sizeof(int));
		p1=realloc(p1,20*sizeof(int));

1. Sample program

```
main()
{
        int *p, n,i;
        printf("How many elements
        :"); scanf("%d",&n);

        p = (int *)malloc(n*sizeof(int));
        /* Accepting data */
        for(i=0; i<n;i++)
          scanf("%d",p+i);

        /* Displaying data */
        for(i=0; i<n;i++)
          printf("%d\t",*(p+i));
}
```

1. Sample program 1 allocates memory dynamically for n integers and accepts and displays the values. Type the sample program 1 given above, execute it. Modify the program to allocate memory such that the allocated memory is initialized to 0.

Set A . Write C programs for the following problems.

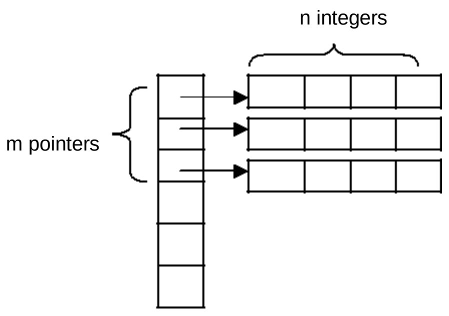
1. Write a program to allocate memory dynamically for n integers such that the memory is initialized to 0. Accept the data from the user and find the range of the data elements.

2. Accept n integers in an array. Copy only the non-zero elements to another array (allocated using dynamic memory allocation). Calculate the sum and average of non-zero elements.

Signature of the instructor		Date	/ /

Set B . Write C programs for the following problems.

1. Accept the number of rows (m) and columns (n) for a matrix and dynamically allocate memory for the matrix. Accept and display the matrix using pointers. (Hint: Use an array of pointers.)



Signature of the instructor		Date	/ /

Set C. Write programs to solve the following problems

1. There are 5 students numbered 1 to 5. Each student appears for different number of subjects in an exam. Accept the number of subjects for each student and then accept the marks for each subject. For each student, calculate the percentage and display. (Hint: Use array of 5 pointers and use dynamic memory allocation)

Signature of the instructor		Date	/ /

Assignment Evaluation Signature

0: Not done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs improvement		5: Well Done	

Exercise 3

Start Date

/ /

To demonstrate strings in C.

You should read the following topics before starting this exercise
1. String literals
2. Declaration and definition of string variables
3. The NULL character
4. Accepting and displaying strings
5. String handling functions

A string is an array of characters terminated by a special character called NULL character(\0). Each character is stored in 1 byte as its ASCII code.

Actions Involving strings	Explanation	Example
Declaring Strings		char message[80];
Initializing Strings		char message[]= { 'H', 'e', 'l', 'l', 'o', '\0' } ; char message [ ] = "Hello";

Accepting Strings	scanf and gets can be used to accept strings	char name[20], address[50];
		printf("\n Enter your name ☺;
		scanf("%s", name);
		printf("\n Enter your address ☺;
		gets(address);

Displaying Strings	printf and puts can be used to display strings.	Printf("\n The name is %s:", name);
		printf("\n The address is :");
		puts(address);

String functions	All string operations are performed using functions in "string.h". Some of the most commonly used functions are
	a. strlen – Returns the number of characters in the string (excluding \0)
	b. strcpy – Copies one string to another
	c. strcmp – Compares two strings. Returns 0 (equal), +ve (first string > second), -ve (first string < second). It is case sensitive
	d. strcmpi – Same as strcmp but ignores case
	e. strcat – Concatenates the second string to the first. Returns the concatenated
		#include <string.h>
		main( )
		{
		char str1[50], str2[50],str3[100];
		printf("\n Give the first string:");
		gets(str1);
		printf("\n Give the second string string:");
		gets(str2);
		if (strlen(str1) == strlen(str2)
		{strcpy(str3, strrev(str1));
		strcat(str3, strupr(str2));
		puts(strupr(str3));
		}
		else
		puts(strlwr(str2);
		}

	string. f. strrev – Reverses a string and returns the reversed string. g. strupr – Converts a string to uppercase. h. strlwr – Converts a string to lowercase	

An array of strings is a two dimensional array of characters. It can be treated as a 1-D array such that each array element is a string.

Actions Involving array of strings	Explanation	Example
Declaring String array	char array[size1][size2];	char cities[4][10]
Initializing String array		char cities[4][10] = { “Pune”, “Mumbai”, “Delhi”, “Chennai”} ;

Sample program-

The following program illustrates how to accept ‘n’ names , store them in an array of strings and search for a specific name.

```
#include <stdio.h>
main( )
{
            char list[10][20];            /*list is an array of 10 strings */
         char name[20];
            int I,n;
            printf(“\n How many names ?:”);
            scanf(“%d”, &n);
            accept(list);

            for (i=0;i<n; i++)
            {
                        printf(“\n Enter name %d,”i);
                        gets(list[i]);
            }

            printf(“\n The names in the list are : \n”);
            for (i=0; i<n; i++)
                        puts(list[i]);

      printf(“\n Enter the name to be searched “);
      gets(name);
            for (i=0; i<n; i++)
                        if(strcmp(list[i],name)==0)
                 {
                    printf(“Match found at position %d”, i);
                    break;
                 }
    if(i==n)
          printf(“Name is not found in the list”);
}
```

1. Write a program to accept two strings str1 and str2. Compare them. If they are equal, display their length. If str1 < str2, concatenate str1 and the reversed str2 into str3. If str1 > str2, convert str1 to uppercase and str2 to lowercase and display. Refer sample code for string functions above.

2. Type the sample program above and execute it. Modify the program to copy the characters after reversing the case. (Hint: First check the case of the character and then reverse it)

Signature of the instructor Date / /

Set A . Write C programs for the following problems.

1. Write a menu driven program to perform the following operations on strings using standard library functions:

Length □Copy □Concatenation □Compare

2. Write a program that will accept a string and character to search. The program will call a function, which will search for the occurrence position of the character in the string and return its position. Function should return –1 if the character is not found in the string.

3. A palindrome is a string that reads the same-forward and reverse. Example: “madam” is a Palindrome. Write a function which accepts a string and returns 1 if the string is a palindrome and 0 otherwise. Use this function in main.

4. Write a program which accepts a sentence from the user and alters it as follows:
Every space is replaced by *, case of all alphabets is reversed, digits are replaced by ?

Signature of the instructor Date / /

Set B . Write C programs for the following problems.

1. Write a program that accepts n strings and displays the longest string.

2. Define two constant arrays of strings, one containing country names (ex: India, France etc) and the other containing their capitals. (ex: Delhi, Paris etc). Note that country names and capital names have a one-one correspondence. Accept a country name from the user and display its capital.
Example: Input: India , Output: Delhi.

Signature of the instructor Date / /

Set C. Write programs to solve the following problems

1. Write a program that accepts a sentence and returns the sentence with all the extra spaces trimmed off. (In a sentence, words need to be separated by only one space; if any two words are separated by more than one space, remove extra spaces)

2. Write a program that accepts a string and displays it in the shape of a kite. Example: "abcd" will be displayed as :

```
   a
  abab
 abcabc
abcdabcd
 abcabc
  abab
   aa
```

3. Write a program that accepts a string and generates all its permutations. For example: ABC, ACB, BAC, BCA, CAB, CBA

Signature of the instructor		Date	/ /

Assignment Evaluation			Signature		
0: Not done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs improvement		5: Well Done	

Start Date	/ /

To demonstrate String operations using pointers

You should read the following topics before starting this exercise
1. How to declare and initialize strings.
2. String handling functions
3. How to create and access an array of strings.
4. Dynamic memory allocation

Sample program :
The following program demonstrates how to pass two strings to a user defined function and copy one string to other using pointers

```
void string_copy (char *t,char *s)
{
          while (*s !='\0')            /* while source string does not end */
          {*t=*s;
                    s++;

                    t++;
          }
          *t ='\0';       /*terminate target string */
           }
main()
{
          char str1[20], str2[20];
          printf("Enter a string :");
          gets(str1);
          string_copy(str2, str1);
          printf("The copied string is :");
          puts(str2);
}
```

1. Type the above sample program and execute the same for different inputs.

Signature of the instructor		Date	/ /

Set A . Write C programs for the following problems.

1. Write a menu driven program to perform the following operations on one string using pointers.

i. Length ii. Reverse iii, Convert to uppercase iv. Convert to lowercase
(Write a separate function for each operation)

2. Write a menu driven program to perform the following operations on two strings using pointers.
i. Copy ii.Concatenation iii. Compare
(Write a separate function for each operation, Copy and Concatenate should return the resulting string)

Signature of the instructor		Date	/ /

Set B. Write C programs for the following problems.

1. Write a program that accepts n words and outputs them in dictionary order. (Use array of pointers and dynamically allocate memory for each word)

2. Write a function which returns the substring of a given string using pointers. Use this function in main. Function prototype: char * substring(char *str, int start, int end);

Signature of the instructor		Date	/ /

Set C. Write programs to solve the following problems
1. Write a function, which displays a given number in words.
For Example: 129 One Hundred Twenty Nine
2019 Two Thousand Nineteen

Signature of the instructor		Date	/ /

Assignment Evaluation Signature

0: Not done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs improvement		5: Well Done	

Exercise 4

Start Date

/ /

Structures and Union in C

You should read the following topics before starting this exercise
1. Concept of structure
2. Declaring a structure
3. Accessing structure members
4. Array of structures
5. Pointer to a structure.
6. Passing structures to functions

A structure is a composition of variables possibly of different data types, grouped together under a single name. Each variable within the structure is called a ‘member’.

Operations performed	Syntax / Description	Example

Declaring a structure	struct structure-name
	{
	    type member-1 ;
	    type member-2;
	    .
	    .
	    type member-n ;
	};
		struct student
		{
		    char name[20];
		    int rollno;
		    int marks;
		};

Creating structure variables	struct structurename variable;	struct student stud1;
Accessing structure members	variable.member	stud1.name stud1.rollno stud1.marks
initializing a structure variable	the initialization values have to be given in {} and in order	struct student stud1 = {“ABCD”,10,95};

Pointer to a structure	struct structure-name * pointer-name;	struct student *ptr;
		    ptr = &stud1;

Accessing members using Pointer	pointer-name -> member-name;	ptr->name; ptr->rollno;
Array of structures	struct structure-name array-name[size];	struct student stud[10];
passing Structures to Functions	return-type function-name (struct structure-name variable);	void display(struct student s);
pass an array of structures to a function	return-type function-name (struct structure-name array[size]);	void display(struct student stud[10]);

Sample Code:

```
#include<stdio.h>
struct student
{char name[20]; int
        rollno;
        int marks[3];
        float perc;
};
void main( )
{
        int i, sum j;
        struct student s[10];
        printf("\n Enter the details of the 10 students \n");
        for (i=0;i<10;i++)
        {
                printf("\n Enter the name and roll number \n");
                scanf("%s%d",s[i].name, &s[i].rollno);
                printf("\n Enter marks for three subjects:");
                sum = 0 ;
                for { j=0;j<3;j++)
                {
                        scanf("%d",&s[i].marks[j]);
                        sum  = sum + s[i].marks[j];
                }
                s[i].perc      =(float)sum/3;
        }
        /*      Display details of students */
        printf("\n\n Name \t Roll no\t Percentage");
        printf("\n===============================\n");
        for(i=0;i<10;i++)
        {
                printf("\n%s\t%d\t%f",s[i].name,s[i].rollno,s[i].perc);
        }
}
```

1. The program in Sample code 1 demonstrates an array of structures of the type student. Type the above program and run it. Modify the program to display the details of the student having the highest percentage.

Signature of the instructor		Date	/ /

Set A . Write C programs for the following problems.

1. Create a structure student (roll number, name, marks of 3 subjects, percentage). Accept details of n students and write a menu driven program to perform the following operations. Write separate functions for the different options.

1. Search
2. Modify
3. Display all student details
4. Display all student having percentage > _____
5. Display student having maximum percentage

2. Create a structure employee (id, name, salary). Accept details of n employees and write a menu driven program to perform the following operations. Write separate functions for the different options.

1. Search by name
2. Search by id
3. Display all
4. Display all employees having salary > _____
5. Display employee having maximum salary

Signature of the instructor		Date	/ /

Set B . Write C programs for the following problems.

1. Create a structure Fraction (numerator, denominator). Accept details of n fractions and write a menu driven program to perform the following operations. Write separate functions for the different options. Use dynamic memory allocation.

1. Display the largest fraction
2. Display the smallest fraction
3. Sort fractions
4. Display all

Note: While accepting fractions, store the fractions in the reduced form.

2. Create a structure Game which has members – name, no of players, names of players. Accept details for one game and display them. Note: Each game can have different number of players. Use array of pointers and dynamic memory allocation for names of players.

Example 1: name- Chess, no of players – 2, names – ABC, DEF
Example 2: name – Cricket , no of players-11, names - __, ___, ___, __, ___ etc

Signature of the instructor		Date	/ /

Set C. Write programs to solve the following problems

1. Accept book details of 'n' books viz, book title, author, publisher and cost. Assign an accession numbers to each book in increasing order. (Use dynamic memory allocation). Write a menu driven program for the following options.

1. Books of a specific author
2. Books by a specific publisher
3. All books having cost >= _____ .
4. Information about a particular book (accept the title)
5. All books.

2. The government of a state wants to collect census information for each city and store the following information : city name, population of the city, literacy percentage. After collecting data for all cities in the state, the government wants to view the data according to :

1. Literacy level
2. Population
3. Details of a specific city.

Write a C program using structures and dynamic memory allocation.

Signature of the instructor		Date	/ /

Assignment Evaluation				Signature	
0: Not done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs improvement		5: Well Done	

Nested Structures and Unions

You should read the following topics before starting this exercise

1. Dynamic memory allocation
2. Structure within a structure
3. Creating and accessing unions

Nested structures: The individual members of a structure can be other structures as well. This is called nesting of structures.

Operations performed	Syntax	Example

Creating a nested structure	struct structure1
	{
	        . . .
	        struct structure2
	    {
	      . . .
	        } variable;
	        . . .
	};
	
	Method 2
	struct structure2
	{
	  . . .
	};
	
	struct structure1
	{
	        . . .
	        struct structure2
	variable;
	    . . .
	};
		struct student
		{
		        int rollno; char name[20];
		        struct date
		    {
		      int dd, mm, yy;
		        } bdate, admdate;
		};
		
		struct date
		{
		  int dd, mm, yy;
		};
		
		struct student
		{
		  int rollno; char name[20];
		  struct date bdate, admdate;
		};

Accessing nested structure members	nested structure members can be accessed using the (.) operator repeatedly.	stud1.bdate.dd, stud1.bdate.mm

self referential structure	A structure containing a pointer to the same structure	struct node
		{
		   int info;
		   struct node *next;
		};

Unions	A union is a variable that
	
	contains multiple members of possibly different data types grouped together under a single name. However, only one of the members can be used at a time. They occupy the same memory area.
		union u
		
		{
		   char a;
		   int b;
		};

Sample Code 1:

Example: The following structure is for a library book with the following details : id, title, publisher, code (1 – Text book, 2 – Magazine, 3 – Reference book). If the code is 1, store no-of-copies. If code = 2, store the issue month name. If code = 3, store edition number. Also store the cost.

```
struct library_book
{
            int id;
           char title[80],publisher[20] ;
            int code;
            union u
            {
                        int no_of_copies;
                        char month[10];
                        int edition;
            }info;
            int cost;
};
void main( )
{
struct library_book book1;
printf("\n Enter the details of the book \n");

printf("\n Enter the id, title and publisher \n");
scanf("%d%s%s",&book1.id, book1.title, book1.publisher); printf("\n
Enter the code: 1-Text Book, 2-Magazine, 3-Reference");
            scanf("%d",book1.code);
switch(book1.code)
{
            case 1: printf("Enter the number of copies :");
                                    scanf("%d",&book1.info.no_of_copies);
                                    break;
      case 2:      printf("Enter the issue month name :");
                                    scanf("%s",book1.info.month);
                                    break;
      case 3:      printf("Enter the edition number:");
                                    scanf("%d",&book1.info.edition);
                                    break;
}
printf("Enter the cost :");
scanf("%d",&book1.cost);

/*          Display details of book */
printf("\n id = %d", book1.id);
printf("\n Title = %s", book1.title);
printf("\n Publisher = %s", book1.publisher);
switch(book1.code)
{
   case 1:         printf("Copies = %d:",
                    book1.info.no_of_copies); break;
case 2:          printf("Issue month name =
                  %s",book1.info.month); break;
 case 3:         printf("Edition number
                  =%d:",book1.info.edition); break;
}
printf("\n Cost = %d", book1.cost);
}
```

1. The sample code 1 given above demonstrates how we can create a variable of the above structure and accept and display details of 1 book. Type the program and execute it. Modify the program to accept and display details of n books.

Signature of the instructor | | Date | / /

Set A . Write C programs for the following problems.

1. Modify the sample program 1 above to accept details for n books and write a menu driven program for the following:

i) Display all reference books
ii)Search magazine according for specific month
iii)Find the total cost of all books (Hint: Use no_of_copies).

2. Modify Sample code 1 in Exercise 16 (Student structure). Add another field – birthdate (dd, mm, yyyy). Accept the details for n students. Accept month and display the students having birthdate in that month. (Hint : Use Nested structure)

Signature of the instructor | | Date | / /

Set B. Write programs to solve the following problems

1. Define structure Item (code, name, price, quantity). Accept details for n items using dynamic memory allocation. Display all details and calculate the total cost of inventory. (Use array of pointers)

Signature of the instructor | | Date | / /

Set C. Write programs to solve the following problems

1. A shop sells electronic items. Each item has an id, company name, code (1-TV, 2-Mobile phones, 3-Camera) and cost. The following additional details are stored for each item.
   - TV - size, type (CRT-1 / LCD- 2 / Plasma-3)
   - Mobile Phone - type (GSM – 1 / CDMA – 2) , model number.
   - Camera – resolution, model number.

The shop wants to maintain a list of all items and perform the following operations for each of the item types:

i) Display all

ii) Search for specific item

iii) Sort according to cost

Signature of the instructor		Date	/ /

Assignment Evaluation Signature

0: Not done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs improvement		5: Well Done	

Exercise 5

Start Date

/ /

File Handling

You should read the following topics before starting this exercise

1. Concept of streams
2. Declaring a file pointer
3. Opening and closing a file
4. Reading and Writing to a text file
5. Command line arguments

Operations performed	Syntax	Example
Declaring File pointer	FILE * pointer;	FILE *fp;
Opening a File	fopen(“filename”,mode); where mode = “r”, “w”, “a”, “r+”, “w+”, “a+”	fp=fopen(“a.txt”, “r”);

Checking for successful open	if (pointer==NULL)	if(fp==NULL)
		exit(0);

Checking for end of file	feof	if(feof(fp))
		printf(“File has ended”);

Closing a File	fclose(pointer);
	fcloseall();
		fclose(fp);

Character I/O	fgetc, fscanf
	fputc, fprintf
		ch=fgetc(fp);
		fscanf(fp, ”%c”,&ch);
		fputc(fp,ch);

String I/O	fgets, fscanf
	fputs, fprintf
		fgets(fp,str,80);
		fscanf(fp, ”%s”,str);

Reading and writing formatted data	fscanf
	fprintf
		fscanf(fp, ”%d%s”,&num,str);
		fprintf(fp, “%d\t%s\n”, num, str);

Random access to files	ftell, fseek, rewind	fseek(fp,0,SEEK_END); /* end of file*/ long int size = ftell(fp);

Sample Code 1

The following program reads the contents of file named a.txt and displays its contents on the screen with the case of each character reversed.

```
#include <stdio.h>
#include <ctype.h>
void main()
{
        FILE * fp;
        fp = fopen(“a.txt”, “r”);
        if(fp==NULL)
        {
                printf(“File opening error”);
                exit(0);
        }
        while( !feof(fp))
        {
                ch = fgetc(fp);
                if(isupper(ch))
                     putchar(tolower(ch));
               else
                    if(islower(ch))
                         putchar(toupper(ch));
                    else
                         putchar(ch);
        }
        fclose(fp);
}
```

Sample Code 2

The following program displays the size of a file. The filename is passed as command line argument.

```
#include <stdio.h>
void main(int argc, char *argv[])
{
        FILE * fp;
        long int size;
        fp = fopen(argv[1], “r”);
        if(fp==NULL)
        {
                printf(“File opening error”);
                exit(0);
        }
        fseek(fp, 0, SEEK_END); /* move pointer to end of file */
        size = ftell(fp);
        printf(“The file size = %ld bytes”, size);
        fclose(fp);
}
```

Sample Code 3

The following program writes data (name, roll number) to a file named student.txt , reads the written data and displays it on screen.

```
#include <stdio.h>
void main()
{
	FILE * fp;
	char str[20]; int num;
	fp = fopen(“student.txt”, “w+”);
	if(fp==NULL)
	{
		printf(“File opening error”);
		exit(0);
	}
	fprintf(fp,“%s\t%d\n”, “ABC”, 1000);
	fprintf(fp,“%s\t%d\n”, “DEF”, 2000);
	fprintf(fp,“%s\t%d\n”, “XYZ”, 3000);

	rewind(fp);
	while( !feof(fp))
	{
		fscanf(fp,“%s%d”, str, &num);
		printf(“%s\t%d\n”, str, num);
	}
	fclose(fp);
}
```

1. Create a file named a.txt using the vi editor. Type the sample program 1 given above and execute the program. Modify the program to accept a character from the user and count the total number of times character occurs in the file.

2. Type the sample program 2 above and execute it. Modify the program to display the last n characters from the file.

3. Type the sample program 3 above and execute it. Modify the program to accept details of n students and write them to the file. Read the file and display the contents in an appropriate manner.

Signature of the instructor		Date	/ /

Set A . Write C programs for the following problems.

1. Write a program to accept two filenames as command line arguments. Copy the contents of the file to the second such that the case of all alphabets is reversed.
2. Write a program to accept a filename as command line argument and count the number of words, lines and characters in the file.

Signature of the instructor		Date	/ /

Set B. Write programs to solve the following problems

1. Write a program to accept details of n students (roll number, name, percentage) and write it to a file named "student.txt". Accept roll number from the user and search the student in the file. Also display the student details having the highest percentage.

2. Write a program which accepts a filename and an integer as command line arguments and encrypts the file using the key. (Use any encryption algorithm).

Signature of the instructor		Date	/ /

Set C . Write C programs for the following problems.

1. A text file contains lines of text. Write a program which removes all extra spaces from the file.

2. Write a menu driven program for a simple text editor to perform the following operations on a file, which contains lines of text.

.i Display the file

.ii Copy m lines from position n to p

.iii Delete m lines from position p

.iv Modify the nth line

.v Add n lines

Signature of the instructor		Date	/ /

Assignment Evaluation Signature

0: Not done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs improvement		5: Well Done	

Exercise 6

Start Date

/ /

Command line arguments and preprocessor directives.

You should read the following topics before starting this exercise

1. Passing arguments from the command line to main
2. Accessing command line arguments
3. File inclusion, macro substitution and conditional compilation directives.
4. Argumented and Nested macros

Preprocessor directives	They begin with a # which must be the first non-space character on the line. They do not end with a semicolon.	
Macro Substitution Directive	# define MACRO value	# define PI 3.142
Argumented macro	# define MACRO(argument) value	# define SQR(x) x*x #define LARGER(x,y) ((x)>(y)?(x):(y))
Nested macro	one macro using another	#define CUBE(x) (SQUARE(x)*(x))

File Inclusion directive	#include <filename>
	#include "filename"
		#include <stdio.h>

Conditional Compilation directive	# if, # else, # elif, # endif #ifdef	#ifdef PI #undef PI #endif

Command Line Arguments	int argc - argument counter
	char *argv[]-argument vector
		void main(int argc, char *argv[])
		{
		printf("There are %d arguments in all", argc);
		for (i=0; i<argc; i++)
		printf("Argument %d =%s",i,argv[i]);
		}

To run a program using command line arguments	Compile the program using cc Execute the program using a.out followed by command line arguments	Example: a.out ABC 20 Here, ABC and 20 are the two command line arguments which are stored in the form of strings. To use 20 as an integer, use function atoi . Example: int num = atoi(argv[2]);

Sample Code 1

```
#define INRANGE(m)  ( m >= 1 && m<=12)
#define NEGATIVE(m) (m<0)
#define ISLOWER(c) (c>='a'&&c<='z')
#define ISUPPER(c) (c>='A'&&c<='Z')
#define ISALPHA(c) (ISUPPER(c)||ISLOWER(c))
#define ISDIGIT(c) (c>='0'&&c<='9')

void main()
{
   int m; char c;
   printf("Enter an integer corresponding to the month");
   scanf("%d",&m);
   if(NEGATIVE(m))
    printf("Enter a positive number");
  else
   if(INRANGE(m))
      printf("You Entered a valid month");

printf("Enter a character :");
c=getchar();
if(ISAPLHA(c))
    printf("You entered an alphabet");
  else
   if(ISDIGIT(c))
      printf("You Entered a digit");
}
```

1. Write a program to display all command line arguments passed to main in the reverse order. Hint: See table above.

2. Sample code 1 above demonstrates the use of argumented and nested macros. Type the program and execute it.

Signature of the instructor		Date	/ /

Set A . Write C programs for the following problems.

1. Write a program to accept three integers as command line arguments and find the minimum, maximum and average of the three. Display error message if invalid number of arguments are entered.
2. Write a program which accepts a string and two characters as command line arguments and replace all occurrences of the first character by the second.

3. Define a macro MAX which gives the maximum of two numbers. Use this macro to find the maximum of n numbers.

Signature of the instructor Date / /

Set B . Write C programs for the following problems.

2. Write a program which accepts a string and an integer (0 or 1) as command line arguments. If 0, sort the string in ascending order and if 1, sort it in descending order. If the user enters invalid number of arguments, display appropriate message. Write a macro for comparing two characters. (Hint – use atoi)

Signature of the instructor Date / /

Set C . Write C programs for the following problems.

1. Create a header file "mymacros.h" which defines the following macros.
i. SQR(x) ii. CUBE(x) - nested iii. GREATER2(x,y) iv. GREATER3 (x,y,z) – nested
v. FLAG (value = 1) (which may or may not be defined)

Include this file in your program. Write a menu driven program to use macros SQR, CUBE, GREATER2 and GREATER3. Your program should run the first two macros if the macro called FLAG has been defined. If it is not defined, execute the other two macros. Run the program twice – with FLAG defined and with FLAG not defined.

Signature of the instructor Date / /

Assignment Evaluation Signature

0: Not done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs improvement		5: Well Done	

Savitribai Phule Pune University,Pune

Relational Database Management System

F. Y. B. Sc. (Computer Science)

SECTION-II

CS -123

SEMESTER II

Name ____

College Name – K.T.H.M.College, Nashik

Roll No. ______________________ Division _____________________

Academic Year- 2019-20

Prepared By
Ms. Reena Bharathi
Ms. Deepali Joshi

Reviewed By
Dr. Shailaja C. Shirwaika

Updated By
Ms.Sunita N.Deore
Ms.Ratna S.Chaudhari
Ms.Asmita L.Taskar
Ms.Priyanka T.Sawale

Table of contents

Introduction

1. About the work book

This workbook is intended to be used by F. Y. B. Sc (Computer Science) students for the Relational Database Management System Assignments in semester II.
The objectives of this book are

1) Defining clearly the scope of the course
2) Bringing uniformity in the way the course is conducted across different colleges
3) Continuous assessment of the course
4) Bring in variation and variety in the experiments carried out by different students in a batch
5) Providing ready reference for students while working in the lab
6) Catering to the need of slow paced as well as fast paced learners

2. How to use this workbook

The workbook is divided into two sections. Section B is related to Relational Database Management System Assignments and they are to be carried out using PostgreSQL in Linux environment

The database Syllabus is divided into seven assignments. The assignments comprise of activities to be carried out on one or more of the six to eight databases. The students have to create, populate database and then perform the activities specified in each of the assignments. A pool of databases will get created as student progresses through the assignments and these databases can be repeatedly used in subsequent assignments. It is mandatory that students create all the databases and also use them in one or other assignment.

Each Assignment has a set A and set B and may contain more sets. Each set contains tasklists related to one database. The tasks are a mix of simple and difficult tasks. Students should complete tasks on specified number of sets. Depending on time availability student should complete more sets.

2.1 Instructions to the students

Please read the following instructions carefully and follow them.

1) Students are expected to carry this book every time they come to the lab for computer science practicals.

2) Students should prepare oneself before hand for the Assignment by reading the relevant material.

3) Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However student should spend additional hours in Lab and at home to cover as many problems as possible given

in this work book.

4) Students should complete the Forms related to Mini Project using pencil and discuss with the instructor/guide and then finalize them before the submission date specified by the instructor/guide

5) Students will be assessed for each exercise on a scale from 0 to 5

i) Not done	0
ii) Incomplete	1
iii) Late Complete	2
iv) Needs improvement	3
v) Complete	4
vi) Well Done	5

2.2. Instruction to the Instructors

1) Explain the assignment and related concepts in around ten minutes using white board if required or by demonstrating the software.

2) Choose appropriate problems to be solved by students.

3) Fill the blanks in queries giving specific values which can vary from student to student.

4) Make sure that students follow the instruction as given above. Students should be encouraged to solve more sets than specified.

4) You should evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.

5) The value should also be entered on assignment completion page of the respective Lab course.

2.3. Instructions to the Lab administrator

You have to ensure appropriate hardware and software is made available to each student.

The operating system and software requirements on server side and also client side are as given below:

1) Server and Client Side - (Operating System) Fedora Core Linux

2) Database server – PostgreSQL 7.0 or above

Section B – Relational Database Management Assignments

SQL is a strongly typed language; implying that any piece of data represented by postgreSQL has an associated data type.

PostgreSQL supports a wide variety of built-in data types, and it also provides an option to the users to add new data types to PostgreSQL , using the CREATE TYPE command. Table lists the data types officially supported by PostgreSQL. Most data types supported by PostgreSQL are directly derived from SQL standards. The following table contains PostgreSQL supported data types for your ready reference

Category	Data type	Description
Boolean	boolean, bool	A single true or false value.
Binary types	bit(n)	An n-length bit string (exactly n)
	bit varying(n), varbit(n)	A variable n-length bit string (upto
Character Types	character(n)	A fixed n-length character string
	char(n)	A fixed n-length character string
	character varying(n)	
	varchar (n)	
	text	A variable length character string
Numeric types	smallint, int2	A signed 2-byte integer
	integer, int, int4	A signed, fixed precision 4-byte
	bigint, int8	A signed 8-byte integer, up to 18
	real, float4	A 4-byte floating point number
	float8, float	An 8-byte floating point number
	numeric(p,s)	An exact numeric type with
Currency	money	A fixed precision, U.S style
	serial	An auto-incrementing 4-byte
Date and time types	date	The calendar date(day, month
	time	The time of day
	time with time zone	the time of day, including time
	timestamp(includes time	
	interval	An arbitrarily specified length

Person –Area Database

Consider the relation Person (pnumber, pname, birthdate, income), Area(aname, area_type). An area can have one or more persons living in it, but a person belongs to exactly one area. The attribute ‘area_type’ can have values either ‘urban’ or ‘rural’. Create the relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).
Insert sufficient number of appropriate records.

Movie Database

Movies(M_name, release_year, budget)
Actor(A_name, role, charges, A_address)
Producer(producer_id, name, P_address)

Each actor has acted in one or more movies. Each producer has produced many movies and each movie can be produced by more than one producers. Each movie has one or more actors acting in it, in different roles.

Create the relations accordingly, so that the relationship is handled properly and the relations are in normalized form(3NF).
Insert sufficient number of appropriate records.

Bank database

Consider the following database maintained by a Bank. The Bank maintains information about its branches, customers and their loan applications.

Following are the tables:
BRANCH (<u>BID</u> INTEGER, BRNAME CHAR (30), BRCITY CHAR (10))
CUSTOMER (<u>CNO</u> INTEGER, CNAME CHAR (20), CADDR CHAR (35), CITY CHAR(20))
LOAN_APPLICATION (<u>LNO</u> INTEGER, LAMTREQUIRED MONEY, LAMTAPPROVED MONEY, L_DATE DATE)

The relationship is as follows:
BRANCH, CUSTOMER, LOAN_APPLICATION are related with ternary relationship.

TERNARY (<u>BID INTEGER, CNO INTEGER, LNO INTEGER</u>).

Student- Teacher database

Consider the following database maintained by a school. The school maintains information about students and the teachers. It also gives information of the subject taught by the teacher.

Following are the tables:
STUDENT (<u>SNO</u> INTEGER, S_NAME CHAR(30), S_CLASS CHAR(10), S_ADDR CHAR(50))
TEACHER (<u>TNO</u> INTEGER, T_NAME CHAR (20), QUALIFICATION CHAR (15),EXPERIENCE INTEGER)
The relationship is as follows:
STUDENT-TEACHER: M-M with descriptive attribute SUBJECT.

Project-Employee database

Consider the database maintained by a company which stores the details of the projects assigned to the employees.

Following are the tables:
PROJECT (<u>PNO</u> INTEGER, P_NAME CHAR(30), PTYPE CHAR(20),DURATION INTEGER)
EMPLOYEE (<u>ENO</u> INTEGER, E_NAME CHAR (20), QUALIFICATION CHAR (15), JOINDATE DATE)
The relationship is as follows:
PROJECT - EMPLOYEE: M-M Relationship , with descriptive attributes as start_date (date), no_of_hours_worked (integer).

Business trip database

Consider the business trip database that keeps track of the business trips of salesman in an office.

Following are the tables:
SALESMAN (<u>SNO</u> INTEGER, S_NAME CHAR (30), START_YEAR YEAR, DEPTNO VARCHAR (10))
TRIP (<u>TNO</u> INTEGER, FROM_CITY CHAR (20), TO_CITY CHAR (20), DEPARTURE_DATE DATE, RETURN DATE)
DEPT (<u>DEPTNO</u> VARCHAR (10), DEPT_NAME CHAR(20))
EXPENSE (<u>EID</u> INTEGER, AMOUNT MONEY)
The relationship is as follows
DEPT-SALESMAN 1 TO M
SALESMAN - TRIP 1 TO M
TRIP - EXPENSE 1 TO 1

Warehouse Database

CITIES(<u>CITY</u> CHAR(20),STATE CHAR(20))
WAREHOUSES(<u>WID</u> INTEGER,WNAME CHAR(30),LOCATION CHAR(20))
STORES(<u>SID</u> INTEGER,STORE_NAME CHAR(20), LOCATION_CITY CHAR(20)) ITEMS(I<u>TEMNO</u> INTEGER,DESCRIPTION TEXT,WEIGHT DECIMAL(5,2), COST DECIMAL(5,2))
CUSTOMER(<u>CNO</u> INTEGER, CNAME CHAR(50),ADDR VARCHAR(50), CU_CITY CHAR(20))
ORDERS(<u>ONO</u> INT,ODATE DATE)

The relationship is as follows
CITIES-WAREHOUSES 1 TO M
WAREHOUSES - STORES 1 TO M
CUSTOMER – ORDERS 1 TO M
ITEMS – ORDERS M TO M relationship with descriptive attribute ordered_quantity
STORES-ITEMS M TO M RELATION with descriptive attribute quantity

Assignment 1 - Stored Procedure

Stored Procedures are collections of SQL commands and program logic stored on the database server. A stored procedure is a set of SQL statements that are stored in the server and executed as a block.

Stored procedures allow most database access logic to be separated from the application logic. One of the indirect benefits of using stored procedures is that application code becomes smaller and easier to understand. Another advantage of stored procedures is that the SQL can be "pre-compiled" increasing the speed of the application.

Syntax:

```
CREATE FUNCTION identifier (arguments) RETURNS type AS '
`DECLARE
   Variable–declarations;
    [……]
BEGIN
    Statement;
    [……..]
END ;
' LANGUAGE 'plpgsql';
```

Creating Dynamic Procedure

The real benefit of stored procedure is when values are passed and received back.

There are three types of parameters.

IN: The default. This parameter is passed to the procedure and can be changed inside the procedure, but remains unchanged outside.

OUT: No value is supplied to the procedure (it is assumed to be null) but it can be modified inside the procedure and is available outside the procedure.

INOUT: The characteristics of both IN and OUT parameters. A value can be passed to the procedure, modified there as well as passed back again.

VERIADIC:A postgreSQL function can accept a variable number of arguments with one condition that all arguments have the same data type.The argument are passed to the function as an array.The function must contain RETURN statement.RETURN clause specifies that data type we are going to return from the function.The return_datatype can be base,composite or domain type or can reference the type of a table column.The return value of function cannot left underline.

Syntax: RETURN *expression;*

Sample Program

```
create or replace function insert_rec()as'
begin
insert into student values(200,'Yash','FYBSc','Nashik');
end;
'language 'plpgsql';
```

```
create or replace function insert_record(in Sno int4,in sname varchar(30),in class varchar(20),
in addr text, out msg text)as'
begin
insert into student values(Sno,Sname,class,addr);
msg:=''Record inserted Successfully'';
end;
'language 'plpgsql';
```

```
create or replace function update_rec(in Studno int4,out msg text)as'
begin
Update student set addr=''Pune'' where Sno=Studno;
msg:=''Record updated Successfully'';
end;
'language 'plpgsql';
```

Self Activity

1. Create a stored procedure named as 'display_message' which will display the message 'Welcome to RDBMS world!!!!'.
2. Create a stored procedure named as 'addrecords' for adding employee records in the table.
3. Create a stored procedure named as 'emp_data' which will retrieve the name and address of employee from the employee table for a given emp_id passed as a parameter.
4. Drop the procedure called as 'display_message'.

Practical Assignment

SET A

1. **Using Bank Database**

   a. Write a procedure to transfer amount of 1000 Rs.from acc_no 10 to acc_no 20.

   b. Write a procedure withdrawal for the following:

      i) Accept balance amount and acc_no for withdrawal as input parameters.
      ii) Check if input amount is less than actual balance of accounts.
      iii) If input amount is less,give the message "Withdrawal allowed from account".Otherwise give the message "Withdrawal not allowed from account".Update the balance .

SET B

1. **Using Bus Transportation Database**

   a. Write a procedure to list the drivers working on a particular date shift wise.Accept the date as an input parameter.

b. List the bus_no and drivers allocated to the buses which are running from ‘Nasik Road’ to ‘CBS’ on date __________.

SET C

1. **Using Student Competition Database**
    a. Write a procedure to count the number of competitions which come under the type ‘Sports’ and no. of competitions which come under the type ‘academics’.
    b. Write a stored procedure which accepts year as input and gives a list of all competitions held in that year.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment 2 - Stored Functions

Stored functions are user defined functions, that are created using the CREATE FUNCTION statement . The functions, thus created, are called stored functions because they are stored as database objects within the PostgreSQL database. The CREATE FUNCTION command names the new function, states its arguments and return type.

Creating a stored function:

```
CREATE FUNCTION identifier (arguments) RETURNS type AS ‘
`DECLARE
   Variable–declarations;
    [……]
BEGIN
    Statement;
    [……..]
END ;
‘ LANGUAGE ‘plpgsql’;
```

Argument Variables:

PL/pgSQL functions can accept argument variables of different types. Function arguments allow a user to pass information into a function, that the function may need.

Each function argument that is received by a function is incrementally assigned to an identifier that begins with the dollar ($) sign and is labeled with the argument number. Thus the identifier $1 is used for the first argument, $2 is used for the second argument and so on an so forth.

PL/pgSQL allows us to create variable aliases. Aliases are created using the ALIAS keyword and it gives us the ability to provide an alternate variable to use when referencing argument variables. All aliases should be declared within the DECLARE section of a block before they are used.

Returning variables:

PL/pgSQL functions must return a value that matches the data type specified as the return type during the function definition. Values are returned from a function using the RETURN statement.

Attributes:

PL/pgSQL provides variable attributes that basically assists or helps the database programmer , when working with database objects. These attributes are %TYPE and %ROWTYPE.

The **%TYPE attribute** : Used to declare a variable with the data type of a referenced database object (a table column). Syntax :
Variable_name table_name.column_name%TYPE ;

The **%ROWTYPE attribute** : Used to declare a PL/pgSQL record variable to have the same structure as the rows in a table , that we specify in the function block. Syntax : Variable_name table_name%ROWTYPE ;

Controlling Program Flow:

Most programming languages provides different ways of controlling the flow of execution, within a program. PL/pgSQL also provides various statements using which a programmer can control the way actions will be executed within a PL/pgSQL function code. PL/pgSQL provides conditional statements and control loops for the same.

	Statement	Syntax
1	The IF/THEN statement	If < Condition> Then statement; [..] Else Statement; [..] End if;
2	Nested If-else if	If <Condition1> Then Statement; [..] Else if <Condition 2> Then Statement; [….] Else if <Condition 3> Then Statement; [……] Else Statement; [….] End if; End if; End if;
3	Loops	LOOP Statement; [………] EXIT [label] WHEN <condition> END LOOP;
4	While Loop	While Condition loop Statement; […….] End loop;
5	For Loop	For Loop–Index In {Reverse} expression1 ..... expression2 Loop Statement; [……..] End loop;
6	For Loop (iterate through a query resultset)	For {record_variable | %rowtype_variable } IN select_statement LOOP Statement; [……….] END LOOP

A recursive Stored function:
Calling a function inside itself is known as recursion and such a function is called as recursive function.

Practice Examples :
a. A simple stored function to demonstrate a basic loop :

```
Create function loop_demo( ) returns integer as ‘
Declare
Cube–num integer := 2;
Begin
Loop
    Cube–num := Cube–num + 2;
    Exit When Cube–num > 20;
End loop;
Return cube-num;
End; ‘ language ‘plpgsql’;
```

b. A stored function that returns the names of students from a given address.

```
Create function for_demo(varchar ) returns text as
‘ Declare
    studaddr alias for $1;
     /* declare a variable to hold student names and set its default value to a new line.
    */ stud_info text := ‘’\n’’;
  --declare a variable to hold rows from the student table.
    row_data student%Rowtype;
Begin
   --iterate through the results of a query.
     For row_data in select * from student
            where s_addr = studaddr order by name
      Loop
         /* insert the name of the matching student into the stud_info variable
          */ stud_info := stud_info || row_data.name || ‘’\n’’;
      end loop;
       -  return the list of students .
        Return stud_info;
    End; ‘ language ‘plpgsql’;
```

c. A recursive Stored function

```
Simple task: calculate number children for parent 1
 1
/|\
7 2 8
/ \
3 4
/ \
5 6
```

```
DROP TABLE test1;
CREATE TABLE test1 ( child int4, parent int4 );
INSERT INTO test1 VALUES ( 2, 1 );
INSERT INTO test1 VALUES ( 7, 1 );
INSERT INTO test1 VALUES ( 8, 1 );
INSERT INTO test1 VALUES ( 3, 2 );
INSERT INTO test1 VALUES ( 4, 2 );
INSERT INTO test1 VALUES ( 5, 4 );
INSERT INTO test1 VALUES ( 6, 4 );
DROP FUNCTION test1 ( int4, int2 );
create or replace function rec_test(int4,int4) returns int4 as
'declare
   rec record;
   cn int4;
 begin
    if $2=100 then
      raise exception ''Incorrect'';
    end if;
select into cn count(*) from test1 where parent=$1;
for rec in select child from test1 where parent=$1
loop
    cn:=rec_test(rec.child,$1+1)+cn;
end loop;
 return cn;
end;
'language 'plpgsql'
```

Solve atleast THREE sets from the sets given below:

SET A

Using Bank Database

a) Write a function that returns the total number of customers of a particular branch. (Accept branch name as input parameter.)

b) Write a function to find the maximum loan amount approved.

SET B:

Using Project-Employee database

a)Write a function to accept project name as input and returns the number of employees working on the project.

b)Write a function to find the number of employees whose date of joining is before ‘03/10/2010’

SET C

Business trip database

a)Write a PL/pgsql function to find a business trip having maximum expenses.

b)Write a PL/pgsql function to count the total number of business trips from ‘Pune’ to ‘Mumbai’.

SET D

Railway Reservation Database

Consider a railway reservation Database of passengers. Passengers reserve berths of a bogie of trains. The bogie capacity of all the bogies of a train is same.

1. TRAIN (TRAIN_NO INT, TRAIN_NAME VARCHAR(20), DEPART_TIME TIME , ARRIVAL_TIME TIME, SOURCE_STN VARCHAR(20) , DEST_STN VARCHAR (20), NO_OF_RES_BOGIES INT , BOGIE_CAPACITY INT)

2. PASSENGER (PASSENGER_ID INT, PASSENGER_NAME VARCHAR(20), ADDRESS VARCHAR(30), AGE INT , GENDER CHAR)
Relationship is as follows:
TRAIN _PASSENGER : M-M with descriptive attributes as follows :
TICKET (TRAIN_NO INT , PASSENGER_ID INT, TICKET_NO INT COMPOSITE KEY, BOGIE_NO INT, NO_OF_BERTHS INT , DATE DATE , TICKET_AMT DECIMAL(7,2),STATUS
CHAR)
The status of a particular berth can be ‘W‘ (waiting) or ‘C‘ (confirmed).

a)Write a PL/pgsql function to calculate the ticket amount paid by all the passengers on 13/5/2009 for all the trains.
b) Write a PL/pgsql function to update the status of the ticket from “waiting“ to “confirm“ for passenger named “Mr. Jadhav”

<u>SET E</u>
Bus transport Database
Consider the following Database of Bus transport system . Many buses run on one route. Drivers are allotted to the buses shiftwise.
Following are the tables:
BUS (BUS_NO INT , CAPACITY INT , DEPOT_NAME VARCHAR(20))
ROUTE (ROUTE_NO INT, SOURCE CHAR(20), DESTINATION CHAR(20), NO_OF_STATIONS INT)
DRIVER (DRIVER_NO INT , DRIVER_NAME CHAR(20), LICENSE_NO INT, ADDRESS CHAR(20), D_AGE INT , SALARY FLOAT)
The relationships are as follows:
BUS_ROUTE : M-1
BUS_DRIVER : M-M with descriptive attributes Date of duty allotted and Shift — it can be 1(Morning) or 2 (Evening).
Constraints :1. License_no is unique. 2. Bus capacity is not null.
a)Write a PL/pgsql function to find out the name of the driver having maximum salary.
b)Write a PL/pgsql function to accept the bus_no and date and print its allotted driver.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment 3 : Cursors

PL/SQL Cursors provide a way to select multiple rows of data from the database and then to process each row individually. Using a cursor, we can traverse up and down a result set and retrieve only those rows which are explicitly requested. Cursors basically help an application to efficiently use a static result set.

Declaring Cursor Variables

All access to cursors in PL/pgSQL goes through cursor variables, which are always of the special data type refcursor. We can declare a cursor variable either as a bound cursor variable or an unbound cursor variable.

a. To create an unbound cursor variable , just declare it as a variable of type refcursor.

b. To create a bound cursor variable, use the following cursor declaration syntax

```
name CURSOR [ ( arguments ) ] FOR query;
```

where arguments, if specified, is a comma-separated list of pairs ‘name datatype’ that define names to be replaced by parameter values in the given query. The actual values to substitute for these names will be specified later, when the cursor is opened (parameterized cursors).

Opening Cursors

Before a cursor can be used to retrieve rows, it must be opened. PL/pgSQL has three forms of the OPEN statement, two of which use unbound cursor variables while the third uses a bound cursor variable.

Syntax for OPEN FOR query

```
OPEN unbound_cursorvar FOR query;
```

Syntax for OPEN FOR EXECUTE

```
OPEN unbound_cursorvar FOR EXECUTE query_string USING expression [, ... ] ];
```

Syntax for Opening a Bound Cursor

```
OPEN bound_cursorvar [ ( argument_values ) ];
```

Using Cursors

Once a cursor has been opened, it can be manipulated with the statements described below.

FETCH

Syntax :

```
FETCH [ direction { FROM | IN } ] cursor INTO target;
```

The direction clause can be any of the following variants :

NEXT, PRIOR, FIRST, LAST, ABSOLUTE count, RELATIVE count, FORWARD, or BACKWARD. Default is NEXT.

MOVE : MOVE repositions a cursor without retrieving any data.

Syntax :

```
MOVE [ direction { FROM | IN } ] cursor;
```

The direction clause can be any of the variants NEXT, PRIOR, FIRST, LAST, ABSOLUTE count, RELATIVE count, ALL, FORWARD [count | ALL], or BACKWARD [count | ALL].

CLOSE : CLOSE closes the portal underlying an open cursor. This can be used to release resources earlier than end of transaction, or to free up the cursor variable to be opened again.

Syntax :

CLOSE cursor;

Practice Examples :

Example 1 :

Consider a relation, Employee (eno, ename, deptno,salary). We write a function, to Define a cursor to print the details of the employee along with commission earned for each employee. Commission is 20% of salary for employees of dept no = 5; its 50% of salary for employees of dept no = 8; its 30% of salary for employees of dept no = 10.

```
Create function cursor_demo( ) returns integer as ‘
Declare
Emp–rec Employee%rowtype
C–emp cursor for Select * from employee;
Comm Number (6,2);
Begin

Loop
Fetch C–emp into emp–rec;
 If emp–rec.deptno = 5 then
    Comm:= emp–rec.salary * 0.2;
Else if emp–rec.deptno = 8 then
    Comm := emp–rec.salary * 0.5;
Else
     If emp–rec.deptno = 10 then
          Comm := emp–rec.salary * 0.3;
            End if;
          End if;
          Endif;
      Raise notice ‘’emp–rec.ename||emp–rec.deptno||emp–rec.salary||comm.
       ‘’; Exit when not found;
End loop;
Close C–emp;
End; ‘ language ‘plpgsql’;
```

Example 2 :
Consider the following relations,
Dept(dno, dname) Employee(eno, dno, ename, salary) and the relation between Dept and Employee is one to many(1:M). Now we write a PL/pgSQL function to print the list of employees, department wise. Here we use 2 cursors, the 1st Cursor retrieves the department Information for each department and the 2nd cursor, to which dept number is sent as a parameter retrieves the employees for that department.

```
Create function parameterizedcursor_demo( ) returns integer
as ‘ Declare
  d–Info cursor for Select * from dept;
E–Info cursor (dnum dept.dno%type) for
Select * From Emplyee Where dno = dnum;
```

```
X number := 0;
Begin
For drec In d–Info
Loop
   Raise notice ‘’ drec.dno || drec.dname’’;
  For erec In E–Info (drec.dno)
    Loop
      Raise notice ‘’erec.eno || erec.ename || erec.salary’’;
      X:=X+1;
    End loop;
Raise notice “Total employees for:||drec.dname||‘is %’|| X’’;
End loop;
Return (X);
End; ‘ language ‘plpgsql’;
```

Solve atleast THREE sets from the sets given below :

SET A

Using the Warehouse database

a) Write a stored function using cursors to accept a city from the user and to list all warehouses in the city.

b) Write a stored function using cursors to find the list of items whose cost is between Rs.5000 to 10000

SET B

Company –Person database

Company(Name varchar(30),address (50),city varchar(20), phone varchar(10), share _value money)

Person(pname varchar(30),pcity varchar (20))

Company_Person are related with M to M relationship with descriptive attribute No_of_shares. Integer

a) Write a stored function using cursors to transfer the shares owned by ‘Sachin ‘ to ‘Rahul’.

b) Write a stored function using cursors to print the total number of distinct investors along with its total invested value.

SET C

Student –Marks database

Student (rollno integer,name varchar(30),address varchar(50),class varchar(10))

Subject(Scode varchar(10),subject name varchar(20))

student and subject are related with M-M relationship with attributes marks scored.

Create a RDB in 3NF for the above and solve the following.

a) Write a stored function using cursors, to accept a address from the user and display the name,subject and the marks of the students staying at that address.

b) Write a stored function using cursors which will calculate total and percentage of each student

<u>SET D</u>

Using railway reservation database

a) Write a stored function using cursors to find the confirmed bookings of all the trains on 18-05-2009

b) Write a stored function using cursors to find the total number of berths not reserved for all the trains on 18-05-2009.

<u>SET E</u>

Using bus driver database

a) Write a stored function using cursors to display the details of a driver, (Accept driver name as input parameter).

b) Write a stored function using cursors to display the details of the buses that run on routes 1,2 (Use two different cursors for route_no = 1 and route_no = 2).

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment 4 : Handling errors and Exceptions

The RAISE statements raise errors and exceptions during a PL/pgSQL function's execution..A Raise statement is also given the level of error it should raise and the string error message it should send to postgreSQL. The string can also be embedded with variables and expressions, that one needs to list along with the error message. The percent (%) sign is used as the place holder for the variables that are inserted into the string.

The syntax of the RAISE statement is as follows :

RAISE level ''message string '' [, identifier [….]];

The three possible values for the RAISE statement's level are as follows:

Value	Explanation
DEBUG	Debug level statements send the specified text as a debug message to the PostgreSQL log.
NOTICE	Notice level statements send the specified text as a Notice;
EXCEPTION	Exception level statements send the specified text as an ERROR. The exception level also causes the current transaction to be aborted.

Practice examples :

Example 1

In this example , the first raise statement gives a debug level message. The second and third raise statements send a notice to the user. The fourth raise statement displays an error and throws an exception, causing the function to end and the transaction to be aborted.

```
 Create function raise_demo ( ) returns integer as
' Declare
    Int_var integer: =1;
Begin
     -- raise a debug level message
      Raise debug ''the raise demo function
began''; Int_var := int_var+1;
--raise a notice stating the change in value of variable
Raise notice '' variable int_var's value is now %
.'',int_var; --raise an exception
Raise exception ''variable % changed. Transaction aborted.'',int_var;
Return 1;
End;
'language 'plpgsql';
```

Example 2

The function given below changes the value of a variable, and on change it raises an exception and stops the function execution.

```
CREATE FUNCTION raise_test () RETURNS
integer AS ' DECLARE
-  Declare an integer variable
for testing. an_integer
INTEGER = 1;
BEGIN
-  Raise a debug level message.
RAISE DEBUG ''The raise_test() function began.'';
an_integer = an_integer + 1;
-  Raise a notice stating that the an_integer variable was changed,
-  then raise another notice stating its new value.
RAISE NOTICE ''Variable an_integer was changed.'';
RAISE NOTICE ''Variable an_integer's value is now
%.'',an_integer; -- Raise an exception.
RAISE EXCEPTION ''Variable % changed. Transaction aborted.'',an_integer;
RETURN 1;
END;
' LANGUAGE 'plpgsql';
```

Solve atleast TWO sets from the sets given below :

SET A

Using Bank Database

a) Write a stored function to print the total number of customers of a particular branch. (Accept branch name as input parameter.) In case the branch name is invalid, raise an exception for the same.

b) Write a stored function to increase the loan approved amount for all loans by 20%. In case the initial loan approved amount was less than Rs 10000, then print a notice to the user, before updating the amount .

SET B

Using Project-Employee database

a)Write a stored function to accept project name as input and print the names of employees working on the project. Also print the total number of employees working on that project. Raise an exception for an invalid project name.

b)Write a stored function to decrease the Hours_worked by 2 hours, for all projects in which employees from department no 2 is working. Raise an exception , in case the hours_worked becomes = 0 , after updation.

SET C

Using Bus transport Database

a)Write a stored function to print the names of drivers working on both shifts on ‘20/04/2014’.

b)Write a stored function to accept the bus_no and date and print its allotted drivers. Raise an exception in case of invalid bus number.

Assignment 5 : Triggers.

A trigger defines a function which occurs before or after , an action on a table. A trigger is implemented through PL/pgSQL, C or any other functional language that PostgreSQL can use to define a function..

A trigger is a PL/pgSQL block that is associated with a table, stored in a database and executed in response to a specific data manipulation event. Triggers can be executed or fired in response to the following events.

a. A row is inserted into table
b. A row in a table is updated.
c. A row in a table is deleted.

Syntax for defining a database trigger

```
Create Trigger trigger–name
{ Before | After} {event [ or event …]} ON
 table–name for each { Row | statement}
execute procedure fucntionname ( arguments) ;
```

A trigger procedure is created with the CREATE FUNCTION command, declaring it as a function with no arguments and a return type of trigger. Special variables created automatically, on call to a trigger function, are as follows :

Variable Name	Purpose and contents
NEW	Holds the new database row for INSERT/UPDATE operations in row-level triggers
OLD	Holds the old database row for UPDATE/DELETE operations in row-level triggers.
TG_NAME	Contains the name of the trigger actually fired.
TG_WHEN	Specifies the trigger timing. Contains a string of BEFORE or AFTER,
TG_LEVEL	Specifies the trigger type. Contains a string of either ROW or STATEMENT
TG_OP	Specifies the trigger operation. Contains a string of INSERT, UPDATE, or DELETE.
TG_RELID	Contains the object ID of the table that caused trigger invocation.
TG_TABLE_NAME	Contains the name of the table that caused the trigger invocation.
TG_TABLE_SCHEMA	Contains the name of the schema of the table that caused trigeer invocation
TG_NARGS	Number of arguments given to the trigger procedure
TG_ARGV[ ]	Contains the arguments for the trigger statement.

Practice Examples :

Example 1
This trigger ensures that any time a row is inserted or updated in the table, the current user name and time are stamped into the row. And it checks that an employee's name is given and that the salary is a positive value.

```
CREATE TABLE employee (
   ename text,
   esalary integer,
   last_date timestamp,
   last_user text
);
 CREATE FUNCTION emp_timestamp() RETURNS trigger AS ‘
   BEGIN
      -  Check that empname and salary are
      given IF NEW.ename IS NULL THEN
         RAISE EXCEPTION 'empname cannot be null';
      END IF;
      IF NEW.esalary IS NULL THEN
         RAISE EXCEPTION '% cannot have null salary', NEW.ename;
      END IF;
      IF NEW.esalary < 0 THEN
         RAISE EXCEPTION '% cannot have a negative salary', NEW.ename;
      END IF;
      -  Remember who changed the payroll when
      NEW.last_date := current_timestamp;
      NEW.last_user := current_user;
      RETURN NEW;
   END;
‘ LANGUAGE ‘plpgsql’;

CREATE TRIGGER emp_stamp BEFORE INSERT OR UPDATE ON
employee FOR EACH ROW EXECUTE PROCEDURE emp_stamp();
```

Solve atleast THREE sets from the sets given below :

SET A
Using Item_supplier Database

Item(itemno integer,Itemname varchar(20),quantity integer)
Supplier(SupplierNo,Supplier name,address,city)
Item_sup(item_no integer ,Supplier_no integer,Rate Money)

Item and supplier are related with many to many relationship .Rate is descriptive attribute.

a. Write a trigger before update on rate field, If the difference in the old rate and new rate to be entered is more than Rs 2000/ . Raise an exception and display the corresponding message

b. Write a trigger before insert or update on rate field, If the rate to be entered is zero then. Raise an exception and display the message “Zero rate not allowed”.

SET B

Student –Marks database
Student (rollno integer,name varchar(30),address varchar(50),class varchar(10))
Subject(Scode varchar(10),subject name varchar(20))
student and subject are related with M-M relationship with attributes marks scored.

a) Write a trigger before deleting a student record from the student table. Raise a notice and display the message “student record is being deleted”

b) Write a trigger to ensure that the marks entered for a student, with respect to a subject is never < 10 and greater than 100.

SET C

News paper database
Newspaper(<u>name</u> varchar(20), language varchar(20),Publisher varchar(20),cost money)
Cities(<u>pincode</u> varchar(6), city varchar(20), state varchar(20))
Newspaper & Cities M to M relationship with descriptive attribute daily_required integer

a) Calculate the length of pincode. Write a trigger which will fire before insert on the cities table which check that the pincode must be of 6 digit. If it is more or less then it display the appropriate message.

b) Write a trigger which will prevent deleting cities from Maharatra state.

SET D

Using Railway Reservation Database
a) create a trigger to validate train arrival time must be less than train departure time.
b) Write a trigger which will be activated before changing the status field in the ticket table and print a message to the user.

SET E

Using Bus Transportation database
a) Define a trigger after insert or update the record of driver if the age is between 18 and 50 give the message “valid entry” otherwise give appropriate message.
b)Define a trigger after delete the record of bus having capacity < 10. Display the message accordingly

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5:Well Done[ ]