RNS INSTITUTE OF TECHNOLOGY

(AICTE Approved, VTU Affiliated and NAAC 'A' Accredited)

UG Programs - CSE, ECE, ISE, EIE and EEE have been Accredited by NBA up to 30.06.2025

DR. VISHNUVARDHAN ROAD, CHANNASANDRA, RR NAGAR POST, BENGALURU – 560 098

ESTD : 2001

An Institute with a Difference

DATA STRUCTURES LABORATORY MANUAL

For Third Semester B.E [VTU/NEP, 2022 syllabus]

Subject Code – BCSL305

NAME :	
BRANCH :	
SECTION :	
USN :	

VISION AND MISSION OF INSTITUTION

Vision

Building RNSIT into a World Class Institution

Mission

To impart high quality education in Engineering, Technology and Management with a Difference, Enabling Students to Excel in their Career by

1. Attracting quality Students and preparing them with a strong foundation in fundamentals so as to achieve distinctions in various walks of life leading to outstanding contributions
2. Imparting value based, need based, choice based and skill based professional education to the aspiring youth and carving them into disciplined, World class Professionals with social responsibility
3. Promoting excellence in Teaching, Research and Consultancy that galvanizes academic consciousness among Faculty and Students
4. Exposing Students to emerging frontiers of knowledge in various domains and make them suitable for Industry, Entrepreneurship, Higher studies, and Research & Development
5. Providing freedom of action and choice for all the Stake holders with better visibility

VISION AND MISSION OF DEPARTMENT

Vision

To be a global leader in imparting Cyber Security education, research & development and empowering young minds to safeguard the digital world

Mission

1. Empower students with Cyber Security essentials through hands-on facilities and a strong ethical foundation.
2. Facilitate collaborative learning environment, teamwork, and global certifications for real-time challenges.
3. Drive innovation in Cyber Security through state-of-the-art research, fostering a culture of exploration through higher learning and entrepreneurship.
4. Promote students to possess qualities of interpersonal, interdisciplinary, leadership, and societal responsibilities.

DATA STRUCTURES LABORATORY-BCSL305
INTERNAL EVALUATION SHEET

EVALUATION (MAX MARKS 50)			
TEST **A**	**REGULAR EVALUATION** **B**	**RECORD** **C**	**TOTAL MARKS** **A+B+C**
20	**20**	**10**	**50**

R1: REGULAR LAB EVALUATION WRITE UP RUBRIC (MAX MARKS 10)				
Sl. No.	**Parameters**	**Good**	**Average**	**Needs improvement**
a.	**Understanding of problem (3 marks)**	Clear understanding of problem statement while designing and implementing the program (3)	Problem statement is understood clearly but few mistakes while designing and implementing program (2)	Problem statement is not clearly understood while designing the program (1)
b.	**Writing program (4 marks)**	Program handles all possible conditions (4)	Average condition is defined and verified. (3)	Program does not handle possible conditions (1)
c.	**Result and documentation (3 marks)**	Meticulous documentation and all conditions are taken care (3)	Acceptable documentation shown (2)	Documentation does not take care all conditions (1)

R2: REGULAR LAB EVALUATION VIVA RUBRIC (MAX MARKS 15)					
Sl. No.	**Parameter**	**Excellent**	**Good**	**Average**	**Needs Improvement**
a.	**Conceptual understanding (10 marks)**	Answers 80% of the viva questions asked (10)	Answers 60% of the viva questions asked (7)	Answers 30% of the viva questions asked (4)	Unable to relate the concepts (1)

R3: REGULAR LAB PROGRAM EXECUTION RUBRIC (MAX MARKS 10)				
Sl. No.	**Parameters**	**Excellent**	**Good**	**Needs Improvement**
a.	**Design, implementation, and demonstration (5 marks)**	Program follows syntax and semantics of C programming language. Demonstrates the complete knowledge of the program written (5)	Program has few logical errors, moderately demonstrates all possible concepts implemented in programs (3)	Syntax and semantics of C programming is not clear (1)
b.	**Result and documentation (5 marks)**	All test cases are successful, all errors are debugged with own practical knowledge and clear documentation according to the guidelines (5)	Moderately debugs the programs , few test case are unsuccessful and Partial documentation (3)	Test cases are not taken care , unable to debug the errors and no proper documentation (1)

R4: RECORD EVALUATION RUBRIC (MAX MARKS 20)					
Sl. No.	**Parameter**	**Excellent**	**Good**	**Average**	**Needs Improvement**
a.	**Documentation (20 marks)**	Meticulous record writing including program, comments and test cases as per the guidelines mentioned (20)	Write up contains program and test cases, but comments are not included (18)	Write up contains only program (15)	Program written with few mistakes (10)

TEST /LAB INTERNALS MARKS (MAX MARKS 20)

TEST #	Write up 6	Execution 28	Viva 6	Sign	Total 40	Avg. 40	Final 20
TEST-1							
TEST-2						$\overline{}$ 40	$\overline{}$ 20

REGULAR LAB EVALUATION (MAX MARKS 10)

Lab program	Date of Execution	Additional programs	Write up (10)	Exen. (10)	Viva (10)	Total 30	Teacher Signature
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
Total Marks		360		30			10

Final Marks obtained from A. Test (20) + B. Regular Evaluation (10) + C. Record (10)	50	Lab in charge:
		HOD:

PREFACE

We have developed this comprehensive laboratory manual on **Data Structures Lab** with two primary objectives: To make the students comfortable with various data structures for solving the problems and to train them in evolving as an efficient C programmer by strengthening their programming abilities.

This material provides the students an exposure to problem solving approaches and solutions to prescribed problems using C programming language. The problems discussed in this manual comprise of a programming solution, viva questions and practicing programming problems constitute an indispensable part of this material.

Our profound and sincere efforts will be fruitful only when students acquire extensive knowledge by reading this manual and apply the concepts learnt apart from the requirements specified in Data Structures Laboratory as prescribed by VTU, Belagavi.

Department of CSE-CY

TABLE OF CONTENTS

DATA STRUCTURES LABORATORY
SEMESTER – III

Course Code	**BCSL305**	**CIE Marks**	50
Number of Contact Hours/Week	0:0:2	**SEE Marks**	50
Total Number of Lab Contact Hours	28	**Exam Hours**	03

Credits – 1

Course Learning Objectives:

This laboratory course enables students to get practical experience in design, develop, implement, analyze and evaluation/testing of

- Dynamic memory management
- Linear data structures and their applications such as stacks, queues and lists
- Non-Linear data structures and their applications such as trees and graphs

Descriptions (if any):

- Implement all the programs in "C " Programming Language and Linux OS.

Programs List:

1.	Develop a Program in C for the following: a) Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), The second field is the date of the Day (A integer), the third field is the description of the activity for a particular day (A dynamically allocated String). b) Write functions create(), read() and display(); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.
2.	Develop a Program in C for the following operations on Strings. a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP) b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR Support the program with functions for each of the above operations. Don't use Built-in functions.
3.	Develop a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX) a. Push an Element on to Stack b. Pop an Element from Stack c. Demonstrate how Stack can be used to check Palindrome d. Demonstrate Overflow and Underflow situations on Stack e. Display the status of Stack f. Exit Support the program with appropriate functions for each of the above operations

4.	Develop a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, % (Remainder), ^ (Power) and alphanumeric operands.
5.	Develop a Program in C for the following Stack Applications a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^ b. Solving Tower of Hanoi problem with n disks

6.	Develop a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX) a. Insert an Element on to Circular QUEUE b. Delete an Element from Circular QUEUE c. Demonstrate Overflow and Underflow situations on Circular QUEUE d. Display the status of Circular QUEUE e. Exit Support the program with appropriate functions for each of the above operations
7.	Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: *USN, Name, Programme, Sem, PhNo* a. Create a SLL of N Students Data by using *front insertion*. b. Display the status of SLL and count the number of nodes in it c. Perform Insertion / Deletion at End of SLL d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack) e. Exit
8.	Develop a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: *SSN, Name, Dept, Designation, Sal, PhNo* a. Create a DLL of N Employees Data by using *end insertion*. b. Display the status of DLL and count the number of nodes in it c. Perform Insertion and Deletion at End of DLL d. Perform Insertion and Deletion at Front of DLL e. Demonstrate how this DLL can be used as Double Ended Queue. f. Exit
9.	Develop a Program in C for the following operationson Singly Circular Linked List (SCLL) with header nodes a. Represent and Evaluate a Polynomial $P(x,y,z) = 6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3$ b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z) Support the program with appropriate functions for each of the above operations
10.	Develop a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers . a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2 b. Traverse the BST in Inorder, Preorder and Post Order c. Search the BST for a given element (KEY) and report the appropriate message d. Exit
11.	Develop a Program in C for the following operations on Graph(G) of Cities a. Create a Graph of N cities using Adjacency Matrix. b. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method

12.	Given a File of N employee records with a set K of Keys (4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Develop a Program in C that uses Hash function H: K →L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.
Laboratory Outcomes: The student should be able to:	

- Analyze various linear and non-linear data structures
- Demonstrate the working nature of different types of data structures and their applications
- Use appropriate searching and sorting algorithms for the give scenario.
- Apply the appropriate data structure for solving real world problems

Conduct of Practical Examination:

- Experiment distribution
  - For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.
  - For laboratories having PART A and PART B: Students are allowed to pick one experiment from PART A and one experiment from PART B, with equal opportunity.
- Change of experiment is allowed only once and marks allotted for procedure to be made zero of the changed part only.
- Marks Distribution *(Need to change in accordance with university regulations)*
  - c) For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks
  - d) For laboratories having PART A and PART B
    - i. Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks
    - ii. Part B – Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

ACKNOWLEDGMENT

A material of this scope would not have been possible without the contribution of many people. We express our sincere gratitude to Dr. R N Shetty, Chairman, RNS Group of Companies and Mr. Satish Shetty, Managing Trustee- R N Shetty Trust for their magnanimous support in all our endeavors.

We are grateful to Dr. M K Venkatesha, Director, Dr. Ramesh Babu H S, Principal for extending their constant encouragement and support.

Department of CSE-CY

Program 1:

Develop a Program in C for the following:

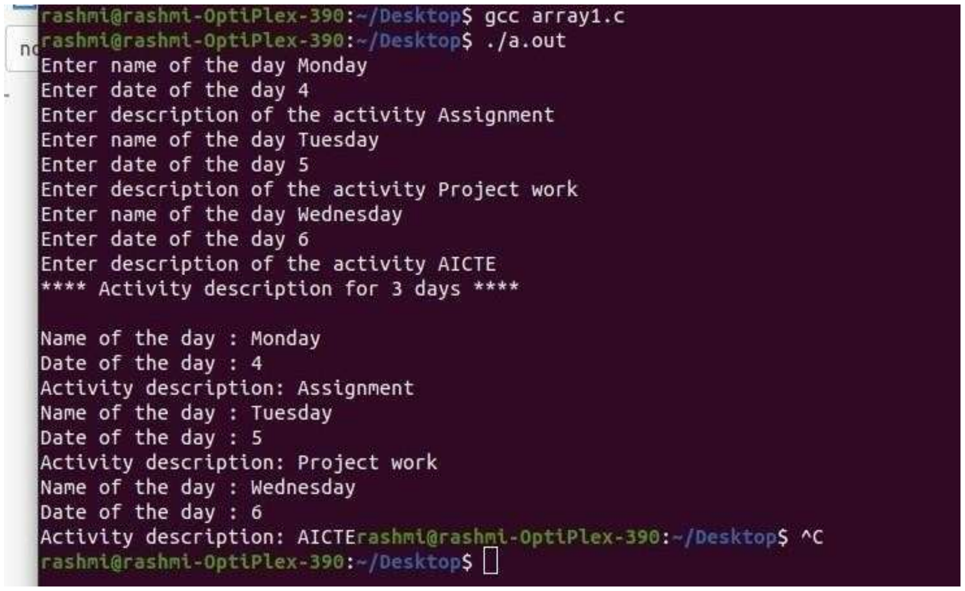
a) Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), The second field is the date of the Day (A integer), the third field is the description of the activity for a particular day (A dynamically allocated String).

b) Write functions create (), read () and display (); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.

```
# include <stdio.h>
# include <string.h>
# include <stdlib.h>
# define DAYS 3
struct activity
{
  char *nday;
  int dday;
  char *desc;
};
typedef struct activity Plan;
Plan* create();
void read(Plan *);
void display(Plan *);

int main()
{
  Plan *cal = NULL;
  cal = create( );

  read(cal);
  display(cal);
}

Plan* create()
{
  Plan *t = (Plan *) malloc(sizeof(Plan)*7);
  if (t == NULL)
  { printf("Sufficient memory not allocated\n"); return 0; }
  return t;
}
```

```
void read(Plan *p)
{

  int i;
  for(i=0;i<DAYS;i++)
  {
          p[i].nday = (char *) malloc(9);
          printf("Enter name of the day "); scanf(" %s",p[i].nday);
          printf("Enter date of the day "); scanf("%d",&(p[i].dday));

          printf("Enter description of the activity ");
          p[i].desc = (char *) malloc(400);
          scanf(" %[^\n]",p[i].desc);
          p[i].desc = (char *) realloc(p[i].desc,strlen(p[i].desc)+1);

  }
}

void display(Plan *p)
{
  int i;
  printf("**** Activity description for %d days ****\n",DAYS);
  for(i=0;i<DAYS;i++)
  {
          printf("\nName of the day : %s", p[i].nday);
          printf("\nDate of the day : %d", p[i].dday);
          printf("\nActivity description %s: ", p[i].desc);
  }
}
```

Output:



```
rashmi@rashmi-OptiPlex-390:~/Desktop$ gcc array1.c
rashmi@rashmi-OptiPlex-390:~/Desktop$ ./a.out
Enter name of the day Monday
Enter date of the day 4
Enter description of the activity Assignment
Enter name of the day Tuesday
Enter date of the day 5
Enter description of the activity Project work
Enter name of the day Wednesday
Enter date of the day 6
Enter description of the activity AICTE
**** Activity description for 3 days ****

Name of the day : Monday
Date of the day : 4
Activity description: Assignment
Name of the day : Tuesday
Date of the day : 5
Activity description: Project work
Name of the day : Wednesday
Date of the day : 6
Activity description: AICTErashmi@rashmi-OptiPlex-390:~/Desktop$ ^C
rashmi@rashmi-OptiPlex-390:~/Desktop$ 
```

Program 2:

Develop a Program in C for the following operations on Strings.

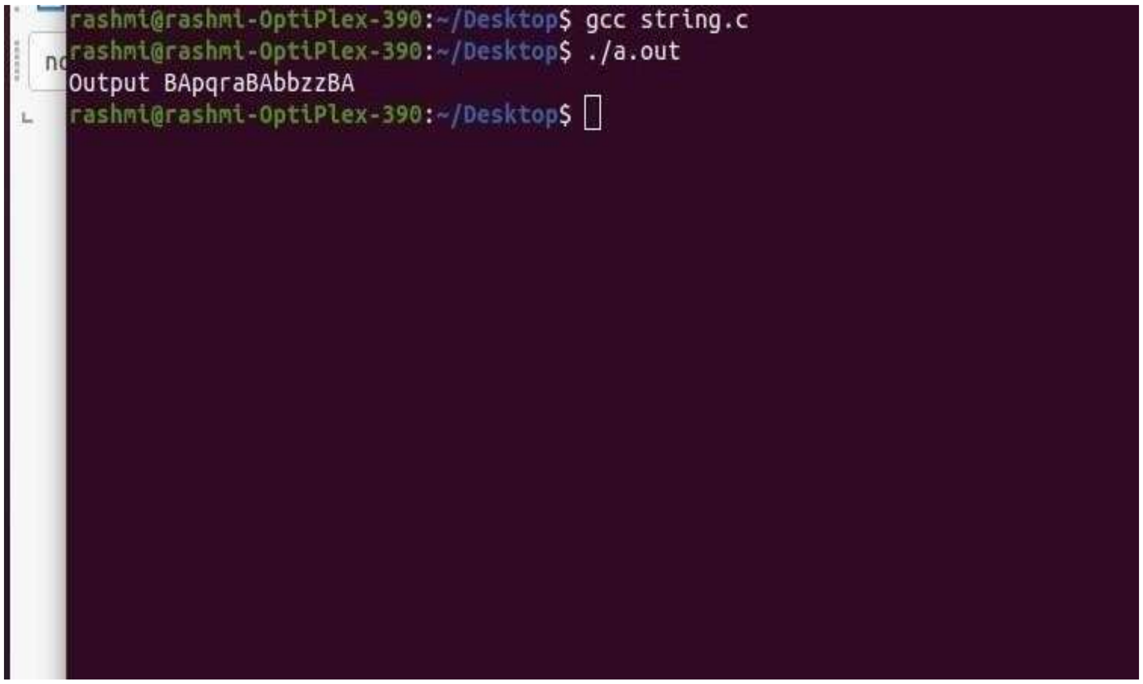
a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)

b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR.

Report suitable messages in case PAT does not exist in STR.
Support the program with functions for each of the above operations. Don't use Built-in functions.

```
# include <stdio.h>
# include <string.h>

int slen(char *);
void replace(char *, char *, char *, char *);

int main()   {
  char T[40]={"bapqrababbzzba"};
  char P[20]={"ba"},REP[20]={"BA"}, FIN[50];

  void replace(char *, char *, char *, char *);
  replace(T,   P, REP, FIN);

  printf("Output %s\n",FIN);
}

int slen(char *s) {
  int len=0;
  for(;s[len] != '\0';len++);
  return len;   }
void replace(char *T, char *P, char *REP, char *FIN)
{
  int k=0,s = slen(T),r=slen(P),e,q=0,z;
  for(;k<s;)
  {
     for(e=0;e<r;e++)
          if (P[e] != T[k+e])
               break;

     if (e==r){
          //printf("Substring found at %d \n",k);
          for(z=0;z<strlen(REP);z++)
          FIN[q++] = REP[z];
          k = k + r;


    }
    else

      FIN[q++] = T[k++];
  }
  FIN[q]='\0';
}
```

Output:



```
rashmi@rashmi-OptiPlex-390:~/Desktop$ gcc string.c
rashmi@rashmi-OptiPlex-390:~/Desktop$ ./a.out
Output BApqraBAbbzzBA
rashmi@rashmi-OptiPlex-390:~/Desktop$
```

Program 3:

Design, Develop and Implement a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)

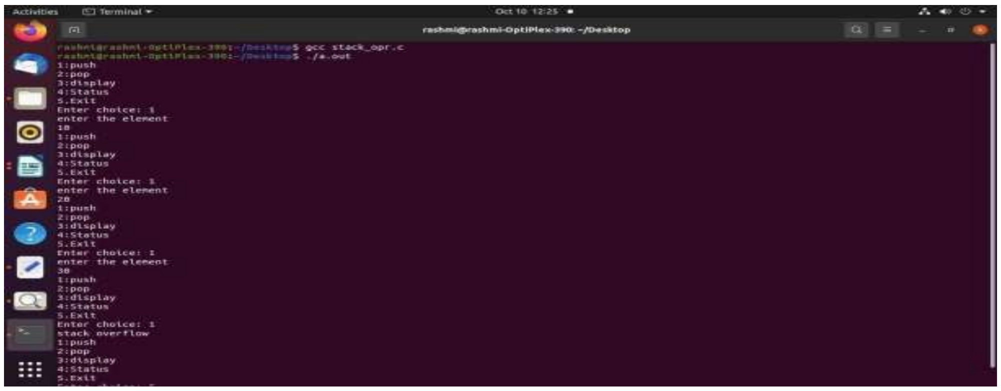
a. Push an Element on to Stack
b. Pop an Element from Stack
c. Demonstrate how Stack can be used to check Palindrome
d. Demonstrate Overflow and Underflow situations on Stack
e. Display the status of Stack
f. Exit

Support the program with appropriate functions for each of the above operations.

```
#include  <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_ELE 4

void push(int [], int *);
void pop(int [], int *);
void display(int [], int *);
void palin( );
void status(int *, int );

int main()
{
        int ch,top=-1;
                int s[MAX_ELE];
        char str[10]=”mom”;
        for(;;)   {
         printf("1:push\n2:pop\n3:palindrome4:display\n5:Status\n6.Ex
         it\n");printf("Enter choice: ");
         scanf("%d",&ch);
         switch(ch)
         {
                case 1: push(s,&top); break;
                case 2:  pop(s,&top);    break;
                case 3: palindrome(str);break;
                case 4:display(s,&top); break;
                case 5: status(s,top); break;
                default: exit(0);
         }
        } }

void status(int *s, int top){
   int used;
   if (top==-1)
        used = 0;
   else
        used = top+1;

   printf("%d locations of the stack are used up\n", used);
   printf("%d locations of the stack are free\n", MAX_ELE - used);
}

 void display(int s[],  int *top)
{
        int i;
        if( (*top) == -1)
        printf("stack empty\n");
        else
        {
          printf("stack elements are\n");
                  printf("TOS is: ");
          for(i=(*top); i>=0; i--)
                printf("%d\n",s[i]);
        }
}

void push(int *s,int *top)
{
        int ele;
        if( (*top)==MAX_ELE-1)
        {  printf("stack overflow\n");      return; }
        (*top)++;
        printf("enter the element\n");      scanf("%d",&ele);
        s[*top]=ele;
}

void pop(int s[],int *top)
{
        if( (*top) == -1)
        printf("stack underflow\n");
        else {
          printf("element popped is\n");
          printf("%d\n",s[*top]);
          (*top)--;
```

```
                    }
          }

Void palindrome (char *s)
{
   int length=strlen(s);
 for(int i=0;i<length;i++)
    push(s[i]);

for(int i=0;i<length;i++)
{

 if(s[top--]!=s[i])
{
 Printf("string is not palindrome");
Exit(0);
}
}
Printf("string is palindrome");
}
```

Output:



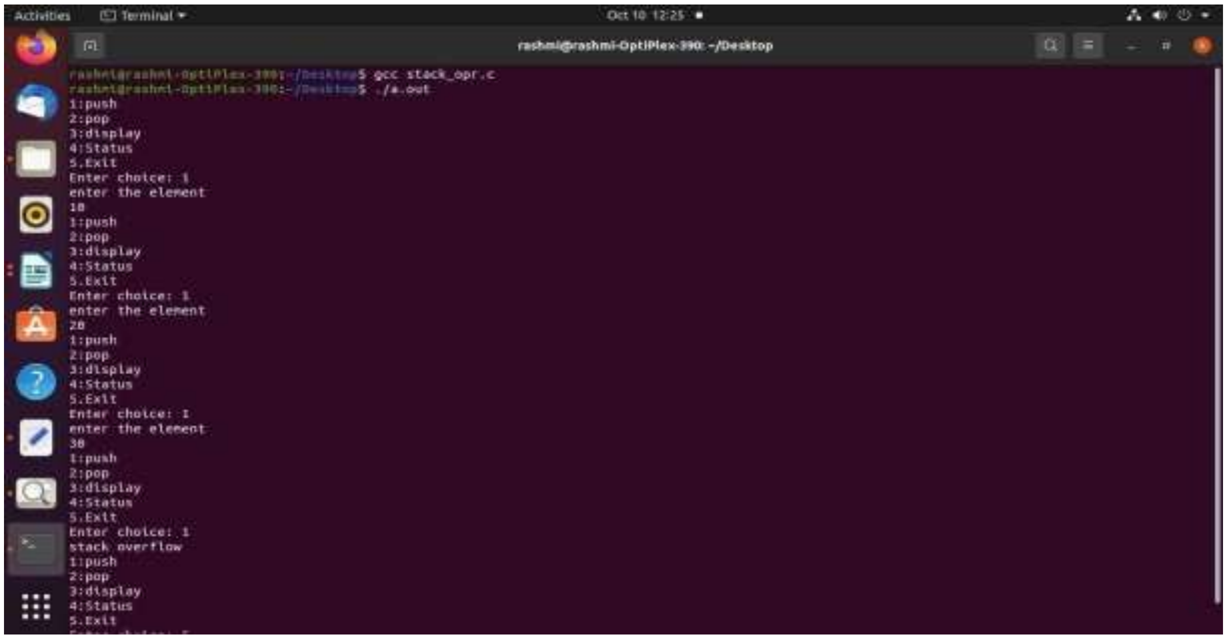
```
rashmi@rashmi-OptiPlex-390:~/Desktop$ gcc stack_opr.c
rashmi@rashmi-OptiPlex-390:~/Desktop$ ./a.out
1:push
2:pop
3:display
4:Status
5.Exit
Enter choice: 1
enter the element
10
1:push
2:pop
3:display
4:Status
5.Exit
Enter choice: 1
enter the element
20
1:push
2:pop
3:display
4:Status
5.Exit
Enter choice: 1
enter the element
30
1:push
2:pop
3:display
4:Status
5.Exit
Enter choice: 1
stack overflow
1:push
2:pop
3:display
4:Status
5.Exit
```

Program 4:

Develop a Program in C for converting an Infix Expression to Postfix expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, % (Remainder), ^ (Power) and alphanumeric operands.

```
#include<stdio.h>
#include<string.h>
#define MAX_ELE 30

int f(char s) {
  switch(s) {
          case '+':
          case '-': return 2;
          case '*':
          case '/': return 4;
          case '$':
          case '^': return 5;
          case '(': return 0;
          case '#': return -1;
          default: return 8;
            } }
int g(char s) {
  switch(s) {
          case '+':
          case '-': return 1;
          case '*':
          case '/': return 3;
          case '$':
          case '^': return 6;
          case '(': return 9;
          case ')': return 0;
          default: return 7;
  } }

int main() {
 char c, s[MAX_ELE]={'#'};
 char inf[MAX_ELE]={"a/b-(c+d)"};
 char pf[MAX_ELE];
  int top=0,i,j=0;  // top = 0  because '#' is already stored in stack
  //printf("enter the infix expression\n");
  //scanf("%s",inf);
  for(i=0;i<strlen(inf);i++)   {
          c= inf[i];
          while( f( s[top] ) > g(c) )
          {
             pf[j]=s[top];
             j++;
            top--;
          }
          if( f( s[top] )  !=  g( c ) )
              s[++top]=c;
          else
              top--;
  }
  for(;s[top]!='#';top--)
          pf[j++]=s[top];
  pf[j]='\0';
  printf("the postfix expression:%s\n",pf);
}
```

Output:



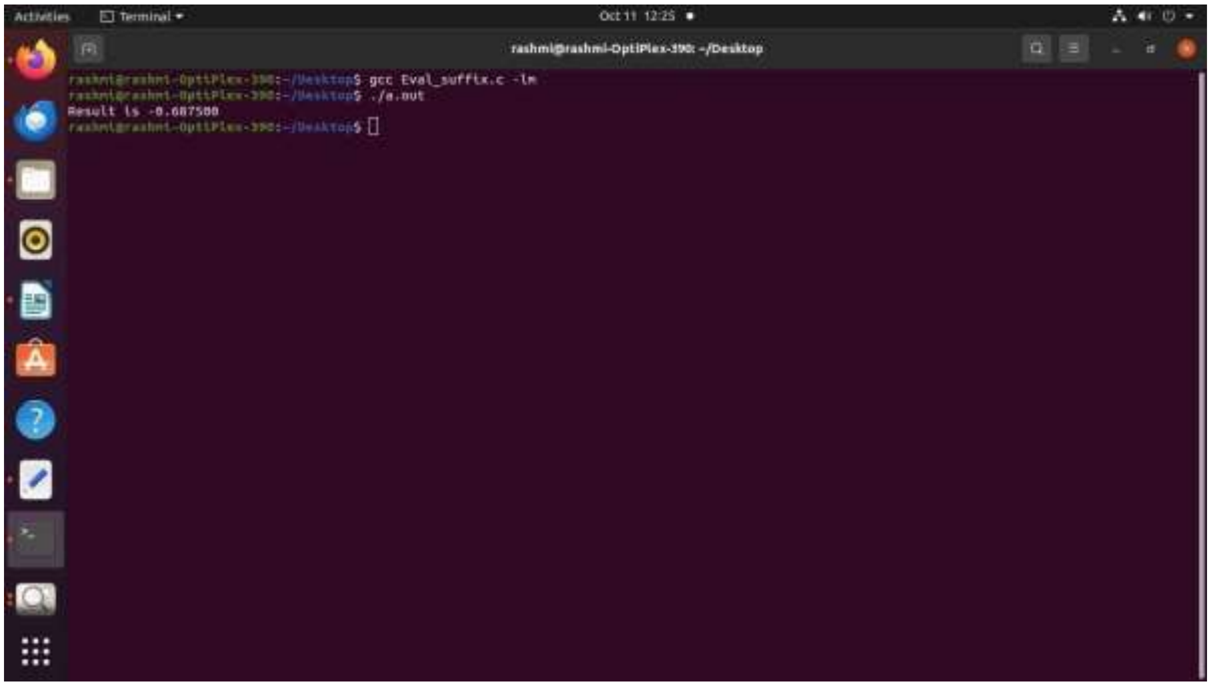
```
rashmi@rashmi-OptiPlex-390:~/Desktop$ gcc stack_opr.c
rashmi@rashmi-OptiPlex-390:~/Desktop$ ./a.out
1:push
2:pop
3:display
4:Status
5.Exit
Enter choice: 1
enter the element
10
1:push
2:pop
3:display
4:Status
5.Exit
Enter choice: 1
enter the element
20
1:push
2:pop
3:display
4:Status
5.Exit
Enter choice: 1
enter the element
30
1:push
2:pop
3:display
4:Status
5.Exit
Enter choice: 1
stack overflow
1:push
2:pop
3:display
4:Status
5.Exit
```

Program 5:

Develop a Program in C for the following Stack Applications
a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %,^

```
# include <stdio.h>
# include <string.h> // for strlen function
# include <math.h>   // for pow function
# include <ctype.h>   // for isdigit function
double compute(char symbol,double op1, double op2)
{
 switch(symbol)
 {
         case '+': return op1 + op2;
         case '-': return op1 - op2;
         case '*': return op1 * op2;
         case '/': return op1 / op2;
         case '$':
         case '^': return pow(op1,op2);
 }
}

int main()
{
 char postfix[20]={"56+437-*/"},symbol;
 double st[20],op1,op2;
 int top=-1,i;

// printf("Enter a valid postfix expression\n");
// scanf("%s",postfix);
 for(i=0;postfix[i] != '\0'; i++)
 {
   if ( isdigit(postfix[i]) )
         st[++top] = postfix[i] - '0';
      else
   {
         op2 = st[top--];
         op1 = st[top--];
         st[++top] = compute(postfix[i],op1,op2);
   }
 }
printf("Result is %lf\n",st[top--]);
}
```

Output

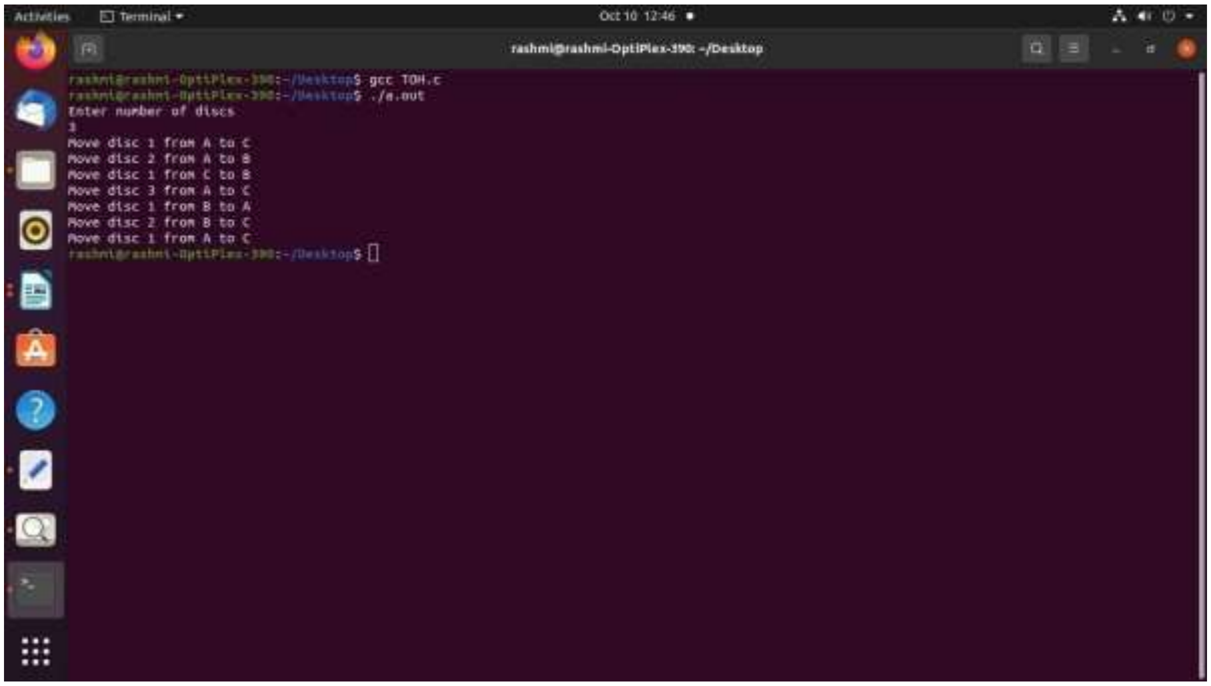


```
rashmi@rashmi-OptiPlex-390:~/Desktop$ gcc Eval_suffix.c -lm
rashmi@rashmi-OptiPlex-390:~/Desktop$ ./a.out
Result is -0.687500
rashmi@rashmi-OptiPlex-390:~/Desktop$ 
```

5b. Solving Tower of Hanoi problem with n disks

```
# include <stdio.h>
void  hanoi (int n, char S, char T, char D)

{
   if (n==0)  return;

   hanoi(n-1, S, D, T);
   printf("Move disc %d from %c to %c\n", n, S, D);
   hanoi(n-1, T, S, D);
}

int main( ) {
  int n;
  printf("Enter number of discs\n");
  scanf("%d",&n);
  hanoi(n , 'A', 'B', 'C');
}
```

Output:



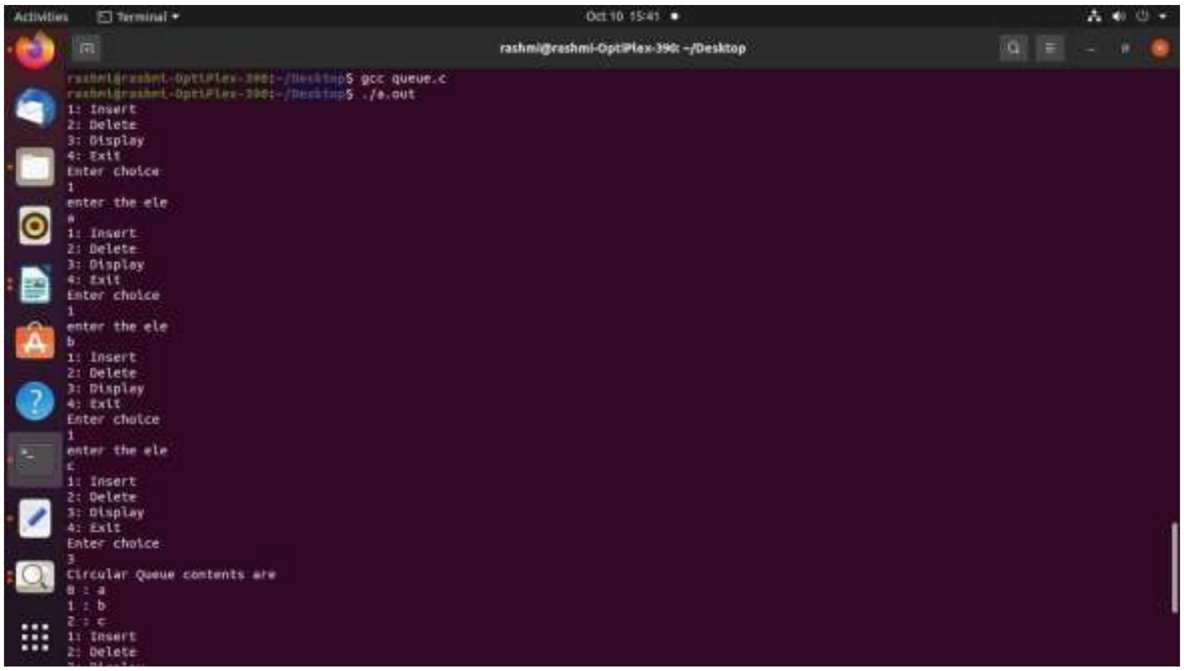
```
rashmi@rashmi-OptiPlex-390:~/Desktop$ gcc TOH.c
rashmi@rashmi-OptiPlex-390:~/Desktop$ ./a.out
Enter number of discs
3
Move disc 1 from A to C
Move disc 2 from A to B
Move disc 1 from C to B
Move disc 3 from A to C
Move disc 1 from B to A
Move disc 2 from B to C
Move disc 1 from A to C
rashmi@rashmi-OptiPlex-390:~/Desktop$ 
```

Program 6:

Develop a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)
a. Insert an Element on to Circular QUEUE
b. Delete an Element from Circular QUEUE
c. Demonstrate Overflow and Underflow situations on Circular QUEUE
d. Display the status of Circular QUEUE
e. Exit
Support the program with appropriate functions for each of the above operations.

```
#include  <stdio.h>
#include <stdlib.h>
#define Max 3
void insert(char [], int *, int *);
void del(char [], int *, int *);
void display(char [], int, int);
int main()
{
        char q[Max];
        int r=-1,f=0,cnt=0;
        int ch;

        while(1)
        {
          printf("1: Insert\n2: Delete\n3: Display\n4: Exit\n");
          printf("Enter choice\n");
          scanf("%d",&ch);
          switch(ch)
         {
           case 1: insert(q,&r,&cnt);  break;
           case 2: del(q,&f,&cnt); break;
           case 3: display(q,f,cnt);    break;
           default: exit(0);
         }   }
  }

void insert(char q[],int *r,int *cnt)
{
        char ele;
        if((*cnt) == Max)
        {
            printf("C Q overflow\n");  return;
    }
    (*r) = ((*r)+ 1 ) % Max;printf("enter the ele\n");scanf(" %c", &ele); q[*r]=ele;
          (*cnt)++;
    }

    void del(char q[],int *f,int *cnt)
    {
          if((*cnt) == 0)
          {printf("C Queue is empty\n"); return;}


          printf("Element deleted from circular queue is %c\n",q[(*f)]);
          (*f) = ((*f) + 1) % Max;
          (*cnt)--;
    }

    void display(char q[], int f, int cnt)
    {
      int i,j;
      if (cnt==0)
      { printf("Circular Queue is empty\n"); return;}
      printf("Circular Queue contents are\n");

      for(i=f,j=0;j<cnt;j++)
      {
            printf("%d : %c\n",i,q[i]);
            i = (i + 1) % Max;
      }
    }
```

Output:



```
rashmi@rashmi-OptiPlex-390:~/Desktop$ gcc queue.c
rashmi@rashmi-OptiPlex-390:~/Desktop$ ./a.out
1: Insert
2: Delete
3: Display
4: Exit
Enter choice
1
enter the ele
a
1: Insert
2: Delete
3: Display
4: Exit
Enter choice
1
enter the ele
b
1: Insert
2: Delete
3: Display
4: Exit
Enter choice
1
enter the ele
c
1: Insert
2: Delete
3: Display
4: Exit
Enter choice
3
Circular Queue contents are
0 : a
1 : b
2 : c
1: Insert
2: Delete
```

Program 7:

Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Programme, Sem, PhNo

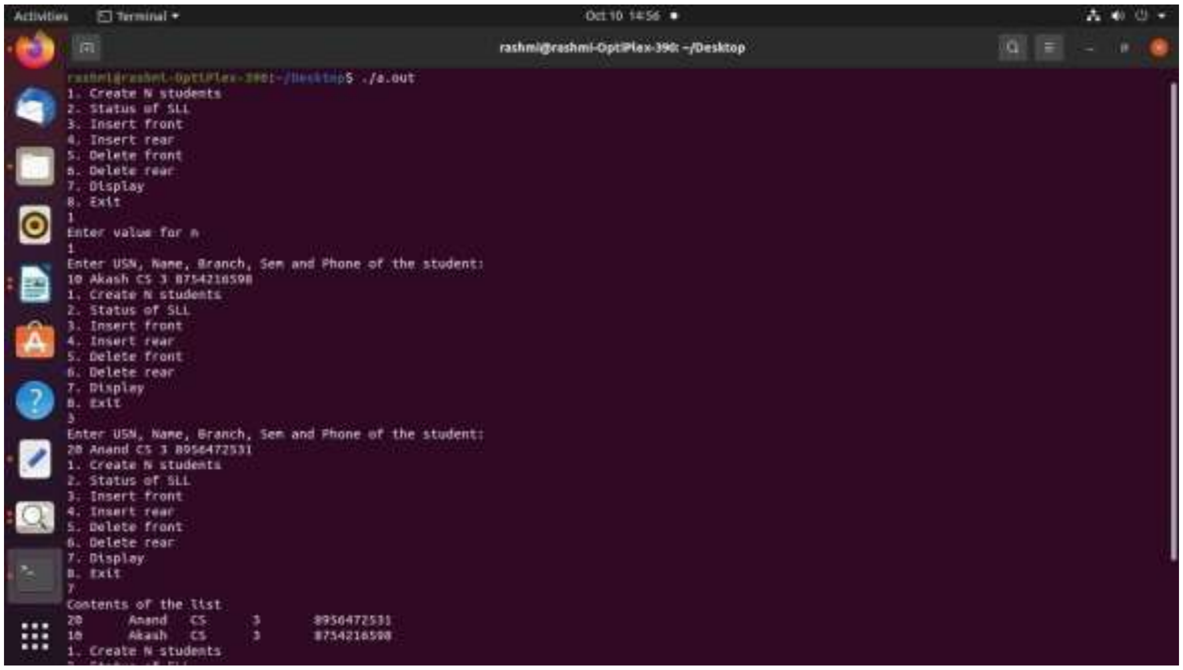
a. Create a SLL of N Students Data by using front insertion.
b. Display the status of SLL and count the number of nodes in it
c. Perform Insertion / Deletion at End of SLL
d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)
e. Exit

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
    char USN[10], name[20], branch[10];
    int sem;
    long int ph;
    struct node *link;
}nd;

nd* create(nd *);
void status(nd *);
nd* ins_front(nd *);
nd* ins_rear(nd *);
nd* del_front(nd *);
nd* del_rear(nd *);
void display(nd *);

int main()
{
  nd * first = NULL;
  int ch;
  for(;;)
  {
          printf("1. Create N students\n2. Status of SLL\n");
          printf("3. Insert front\n4. Insert rear\n5. Delete front\n");
          printf("6. Delete rear\n7. Display\n8. Exit\n");
          scanf("%d", &ch);
          switch(ch)
          {
            case 1: first = create(first);break;
            case 2: status(first); break;
```

```
                case 3: first = ins_front(first); break;
                case 4: first = ins_rear(first); break;
                case 5: first = del_front(first); break;
                case 6: first = del_rear(first); break;
                case 7: display(first); break;
                case 8: exit(0);
              }
          }
        }

        nd * del_front(nd *f)
        {
          nd *t;
          if (f==NULL)
          {
                printf("SLL is empty\n");
                return NULL;
          }
          printf("Information to be deleted is...\n");
    printf("%s\t%s\t%s\t%d\t%ld\n",(f->USN),(f->name),(f->branch),(f->sem),(f->ph));

          t = f->link;
          free(f);
          return t;    }

        nd * del_rear(nd *f)  {
          nd *t,*p;
          if (f==NULL)    {
                printf("SLL is empty\n");
                return NULL;
          }

          for(p=NULL,t=f;t->link!=NULL;p=t,t=t->link);
          printf("Information to be deleted is...\n");
    printf("%s\t%s\t%s\t%d\t%ld\n",(t->USN),(t->name),(t->branch),(t->sem),(t->ph));
          free(t);

          if (p!=NULL)
          {       p->link=NULL; return f; }
          else
                return NULL;
        }
        nd * ins_rear(nd * f)
        {
          nd *p=f;
          nd *t=(nd*)malloc(sizeof(nd));
          t->link=NULL;
          printf("Enter USN, Name, Branch, Sem and Phone of the student:\n");
    scanf("%s%s%s%d%ld",(t->USN),(t->name),(t->branch), &(t->sem), &(t->ph));

          if (f==NULL) return t;

          for(;p->link!=NULL; p=p->link);
          p->link=t;
          return f;
        }

        void status(nd *f)
        {
          int cnt=0;
          if (f==NULL)
          {
                printf("SLL is empty\n");
                return;
          }

          for(;f!=NULL;f=f->link,cnt++);
          printf("Number of nodes in SLL is %d\n",cnt);
        }

        nd* create(nd *f)
        {
          int n,i;
          printf("Enter value for n\n"); scanf("%d",&n);
          for(i=0;i<n;i++)
           f = ins_front(f);
          return f;  }
        nd* ins_front(nd *f)
        {
            nd *t=(nd*)malloc(sizeof(nd));
            printf("Enter USN, Name, Branch, Sem and Phone of the student:\n");
     scanf("%s%s%s%d%ld",(t->USN),(t->name),(t->branch), &(t->sem), &(t->ph));
```

```
            t->link = f;
            return t;
    }

    void display(nd *f)
    {
     if (f==NULL)
     {
       printf("Contents of SLL are empty\n");
       return;
     }
     printf("Contents of the list\n");
     while(f!=NULL)
     {
printf("%s\t%s\t%s\t%d\t%ld\n",(f->USN),(f->name),(f->branch),(f->sem),(f->ph));
       f = f->link;
     }
    }
```

Output:



```
rashmi@rashmi-OptiPlex-390:~/Desktop$ ./a.out
1. Create N students
2. Status of SLL
3. Insert front
4. Insert rear
5. Delete front
6. Delete rear
7. Display
8. Exit
1
Enter value for n
1
Enter USN, Name, Branch, Sem and Phone of the student:
10 Akash CS 3 8754216598
1. Create N students
2. Status of SLL
3. Insert front
4. Insert rear
5. Delete front
6. Delete rear
7. Display
8. Exit
3
Enter USN, Name, Branch, Sem and Phone of the student:
20 Anand CS 3 8956472531
1. Create N students
2. Status of SLL
3. Insert front
4. Insert rear
5. Delete front
6. Delete rear
7. Display
8. Exit
7
Contents of the list
20      Anand   CS      3       8956472531
10      Akash   CS      3       8754216598
1. Create N students
```

Program 8:

Develop a menu driven Program in C for the following operations on Doubly Linked List(DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo

a. Create a DLL of N Employees Data by using end insertion.
b. Display the status of DLL and count the number of nodes in it
c. Perform Insertion and Deletion at End of DLL
d. Perform Insertion and Deletion at Front of DLL
e. Demonstrate how this DLL can be used as Double Ended Queue.
f. Exit

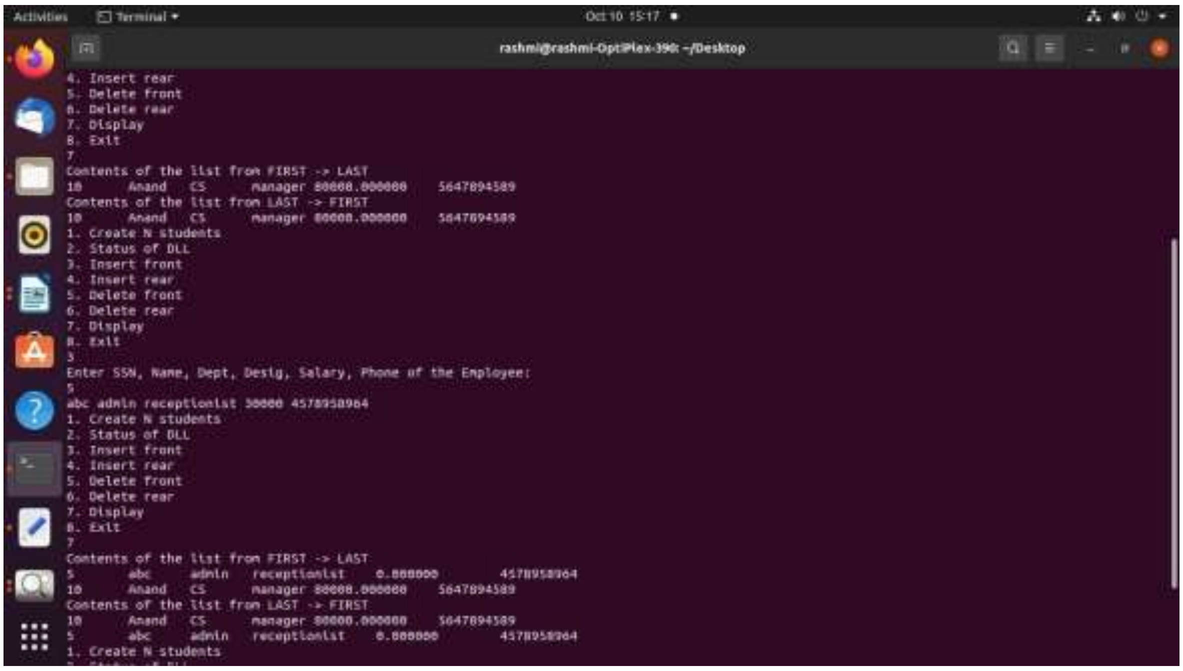
```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
  char SSN[10], name[20], dept[30],desig[30];
  float sal;
  long int ph;
  struct node *llink, *rlink;
}nd;

nd* create(nd *);
void status(nd *);
nd* ins_front(nd *);
nd* ins_rear(nd *);
nd* del_front(nd *);
nd* del_rear(nd *);
void display(nd *);

int main()
{
  nd * first = NULL;
  int ch;
  for(;;)
  {
          printf("1. Create N students\n2. Status of DLL\n");
          printf("3. Insert front\n4. Insert rear\n5. Delete front\n");
          printf("6. Delete rear\n7. Display\n8. Exit\n");
          scanf("%d", &ch);
          switch(ch)
          {
            case 1: first = create(first);break;
            case 2: status(first); break;
            case 3: first = ins_front(first); break;
            case 4: first = ins_rear(first); break;
```

```
                case 5: first = del_front(first); break;
                case 6: first = del_rear(first); break;
                case 7: display(first); break;
                case 8: exit(0);
               }
     }
    }

    nd * del_front(nd *f)
    {
      nd *t;
      if (f==NULL)  {
            printf("DLL is empty\n");
            return NULL;  }

      printf("Information to be deleted is...\n");
 printf("%s\t%s\t%s\t%s\t%f\t%ld\n",f->SSN,f->name,f->dept,f->desig,f->sal,f->ph);

      t = f->rlink;
      if (t)
            t->llink = NULL;
      free(f);
      return t;
    }
    nd * del_rear(nd *f)
    {
      nd *t,*p;
      if (f==NULL)
      {
            printf("DLL is empty\n");
            return NULL;
      }

      for(p=NULL,t=f;t->rlink!=NULL;p=t,t=t->rlink);

      printf("Information to be deleted is...\n");
printf("%s\t%s\t%s\t%s\t%f\t%ld\n",t->SSN,t->name,t->dept,t->desig,f->sal,t->ph);
      free(t);
      if (p!=NULL)
      {
            p->rlink=NULL;
            return f;
      }
```

```
else
return NULL;
}


nd * ins_rear(nd * f)
{
  float sal;
  nd *p=f;
  nd *t=(nd*)malloc(sizeof(nd));
  t->rlink=t->llink=NULL;

  printf("Enter SSN, Name, Dept, Desig, Salary, Phone of the Employee:\n");
scanf("%s%s%s%s%f%ld",t->SSN,t->name,t->dept, t->desig,&(t->sal),&(t-ph));

  if (f==NULL) return t;

  for(;p->rlink!=NULL; p=p->rlink);
  p->rlink=t;
  t->llink=p;

  return f;
}
void status(nd *f)
{
  int cnt=0;
  if (f==NULL)
  {
        printf("DLL is empty\n");
        return;
  }

  for(;f!=NULL;f=f->rlink,cnt++);
  printf("Number of nodes in SLL is %d\n",cnt);
}

nd* create(nd *f)
{
  int n,i;
  printf("Enter value for n\n"); scanf("%d",&n);
  for(i=0;i<n;i++)
   f = ins_rear(f);
  return f;
}
```

```
        nd* ins_front(nd *f)
        {
          float sal;
          nd *t=(nd*)malloc(sizeof(nd));
          t->rlink = t->llink = NULL;

          printf("Enter SSN, Name, Dept, Desig, Salary, Phone of the Employee:\n");
   scanf("%s%s%s%s%f%ld",t->SSN,t->name,t->dept,t->desig,&(t->sal),&(t->ph));

          t->sal = sal;

          t->rlink = f;
          if (f!=NULL)
                  f->llink=t;
          return t;
        }

        void display(nd *f)
        {
         if (f==NULL)
         {
           printf("Contents of DLL are empty\n");
           return;
         }
         printf("Contents of the list from FIRST -> LAST\n");
         while(f->rlink!=NULL)
         {
printf("%s\t%s\t%s\t%s\t%f\t%ld\n",f->SSN,f->name,f->dept,f->desig,f->sal,f->ph);
           f = f->rlink;
         }

printf("%s\t%s\t%s\t%s\t%f\t%ld\n",f->SSN,f->name,f->dept,f->desig,f->sal,f->ph);

         printf("Contents of the list from LAST -> FIRST \n");
         while(f != NULL)
         {
printf("%s\t%s\t%s\t%s\t%f\t%ld\n",f->SSN,f->name,f->dept,f->desig,f->sal,f->ph);
            f = f->llink;
         }
        }
```

Output:



```
4. Insert rear
5. Delete front
6. Delete rear
7. Display
8. Exit
7
Contents of the list from FIRST -> LAST
10      Anand   CS      manager 80000.000000    5647894589
Contents of the list from LAST -> FIRST
10      Anand   CS      manager 80000.000000    5647894589
1. Create N students
2. Status of DLL
3. Insert front
4. Insert rear
5. Delete front
6. Delete rear
7. Display
8. Exit
3
Enter SSN, Name, Dept, Desig, Salary, Phone of the Employee:
5
abc admin receptionist 30000 4578958964
1. Create N students
2. Status of DLL
3. Insert front
4. Insert rear
5. Delete front
6. Delete rear
7. Display
8. Exit
7
Contents of the list from FIRST -> LAST
5       abc     admin   receptionist    0.000000        4578958964
10      Anand   CS      manager 80000.000000    5647894589
Contents of the list from LAST -> FIRST
10      Anand   CS      manager 80000.000000    5647894589
5       abc     admin   receptionist    0.000000        4578958964
1. Create N students
```

Program 9:

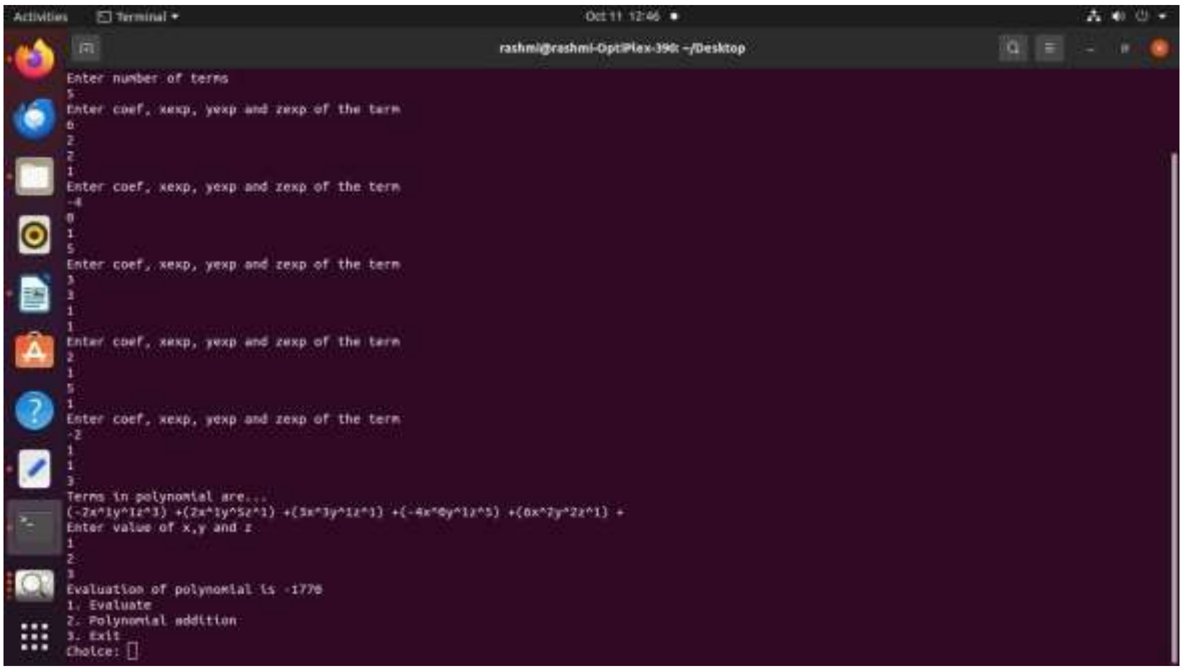
Develop a Program in C for the following operationson Singly Circular Linked List (SCLL) with header nodes.

a. Represent and Evaluate a Polynomial P(x,y,z) = 6x 2 y 2 z-4yz 5 +3x 3 yz+2xy 5 z-2xyz 3
b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the resultin POLYSUM(x,y,z)
Support the program with appropriate functions for each of the above operations.

```
# include  <stdio.h>
# include <stdlib.h>
# include  <math.h>
struct term
{
  int coef, xexp, yexp, zexp;
  struct term *link;
};
typedef struct term tm;

void create(tm *);
void display(tm *);
void evaluate(tm *);
void add( tm *, tm *, tm *);
int main()
{
  tm p1={.link=&p1},p2={.link=&p2},p3={.link=&p3};
  int ch;
  while(1)
  {
          printf("1. Evaluate\n2. Polynomial addition\n3. Exit\n");
          printf("Choice: "); scanf("%d",&ch);
          switch(ch)
          {
          case 1: if (p1.link != &p1) p1.link = &p1;
                    create(&p1);
                    printf("Terms in polynomial are...\n");
                    display(&p1);
                    evaluate(&p1);
                    break;
          case 2: if (p1.link != &p1) p1.link = &p1;
                    if (p2.link != &p2) p2.link = &p2;
                    if (p3.link != &p3) p3.link = &p3;
                    create(&p1);  create(&p2);
                    printf("Terms in poly1 are...\n"); display(&p1);
                    printf("Terms in poly2 are...\n"); display(&p2);
                    add(&p1,&p2,&p3);
                    printf("Resultant polynomial...\n"); display(&p3);
                    break;
              case 3: return 0;
              }
     }
  }

  int compare(tm * p, tm *q)
  {
   if ( (p->xexp == q->xexp) && (p->yexp == q->yexp) && (p->zexp == q-
  >zexp) )
            return 1;
    return 0;
  }

  void attach(int s, tm *p, tm *r)
  {
      tm *t = (tm *) malloc(sizeof(tm));
      t->coef = (s!=0)?s:p->coef;      t->xexp = p->xexp;    t->yexp = p->yexp;
      t->zexp = p->zexp;       t->link =r->link;
      r->link = t;
  }
  void delete(tm *p, tm *tp)
  {
      tm *prev = p, *next = p->link;
      while(next!=tp) {
        prev = next;
        next = next->link; }

      prev->link = next->link;
  }

  void add(tm *p, tm *q, tm *r)
  {
    int val;
    tm *t = (tm *) malloc(sizeof(tm));
    tm *tp=p->link, *tq, *tr=NULL;
    while(tp != p)
    {
       tq = q->link;
       while (tq != q)
       {
        val = compare(tp,tq);

         if (val)
               break;
              tq = tq->link;
           }
           switch(val)
           {
             case 0: attach(0,tp,r);
                         tp = tp->link;
                         delete(p,tp); break;

              case 1: val = tp->coef + tq->coef;
                          attach(val,tp,r);
                          delete(p,tp);  delete(q,tq);
                          tp = tp->link; tq = tq->link;
                          break;
            }
          }

          tq=q->link;
          while (tq != q)
          {
              attach(0,tq,r);
              tq = tq ->link;
          }
        }

        void evaluate(tm *p)
        {
          int x,y,z,res=0; tm *t;
          printf("Enter value of x,y and z\n");
          scanf("%d%d%d",&x,&y,&z);

          for(t=p->link; t!=p ; t=t->link)
                 res=res+t->coef*pow(x,t->xexp)*pow(y,t->yexp)*pow(z,t->zexp);
          printf("Evaluation of polynomial is %d\n",res);
        }
        void display(tm *p)
        {
          tm *t;
          if (p->link == p)
          {
                 printf("SCLL is empty\n");
```

```
return;
  }
  for(t=p->link; t!=p ; t=t->link)
printf("(%dx^%dy^%dz^%d) +",t->coef,t->xexp,t->yexp,t->zexp);
printf("\n");
}

void create(tm *p)
{
  int n,i;
  tm *t;
  printf("Enter number of terms\n");  scanf("%d",&n);

  for(i=0;i<n;i++)
  {
          t = (tm *) malloc(sizeof(tm));

          printf("Enter coef, xexp, yexp and zexp of the term\n");
          scanf("%d%d%d%d",&(t->coef),&(t->xexp),&(t->yexp),&(t->zexp));

          t->link = p->link;
          p->link = t;
  }
}
```

Output:



```
Enter number of terms
5
Enter coef, xexp, yexp and zexp of the term
6
2
2
1
Enter coef, xexp, yexp and zexp of the term
-4
0
1
5
Enter coef, xexp, yexp and zexp of the term
3
3
1
1
Enter coef, xexp, yexp and zexp of the term
2
1
5
1
Enter coef, xexp, yexp and zexp of the term
-2
1
1
3
Terms in polynomial are...
(-2x^1y^1z^3) +(2x^1y^5z^1) +(3x^3y^1z^1) +(-4x^0y^1z^5) +(6x^2y^2z^1) +
Enter value of x,y and z
1
2
3
Evaluation of polynomial is -1770
1. Evaluate
2. Polynomial addition
3. Exit
Choice: 
```

Program 10:

Develop a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers .
a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2
b. Traverse the BST in Inorder, Preorder and Post Order
c. Search the BST for a given element (KEY) and report the appropriate message
d. Exit

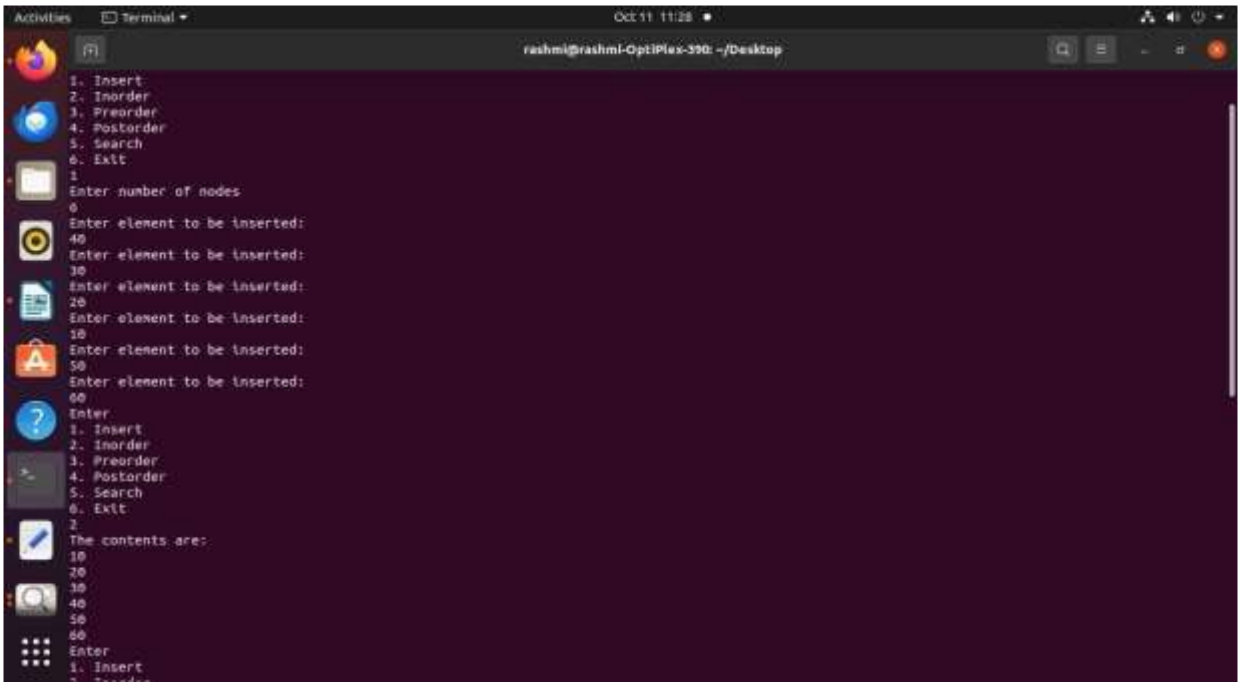
```
#include<stdio.h>
#include<stdlib.h>
typedef struct tree
{
        int data;
        struct tree *rlink, *llink;
}TNODE;
TNODE * getnode();
TNODE * insert(TNODE * root);
void inorder(TNODE  *  root);
void preorder(TNODE * root);
void postorder(TNODE * root);
int search(TNODE * root, int key);
void main()
{
        TNODE *root= NULL;
        int choice, ele, key, flag;
for(;;)
        {
printf("Enter\n1. Insert\n2. Inorder\n3. Preorder\n4. Postorder\n5. Search\n6. Exit\n");
                scanf("%d", &choice);
                switch(choice)
                {
                        case 1:
                                root= insert(root);
                                break;
                        case 2: if(root==NULL)
                                        printf("Tree is empty\n");
                                else
                                {
                                        printf("The contents are:\n");
                                        inorder(root);
                                }
                                break;

                        case 3: if(root==NULL)
                                        printf("Tree is empty\n");
```

```
            else
            {
                    printf("The contents are:\n");
                    preorder(root);
            }
            break;

    case 4: if(root==NULL)
                    printf("Tree is empty\n");
            else
            {
                    printf("The contents are:\n");
                    postorder(root);
            }
            break;

    case 5: printf("Enter the node to be searched:\n");
            scanf("%d", &key);
            flag= search(root, key);
            if(flag==-1)
                    printf("Unsuccessful search!!!\n");
            else
                    printf("Successful search!!!\n");
            break;
```

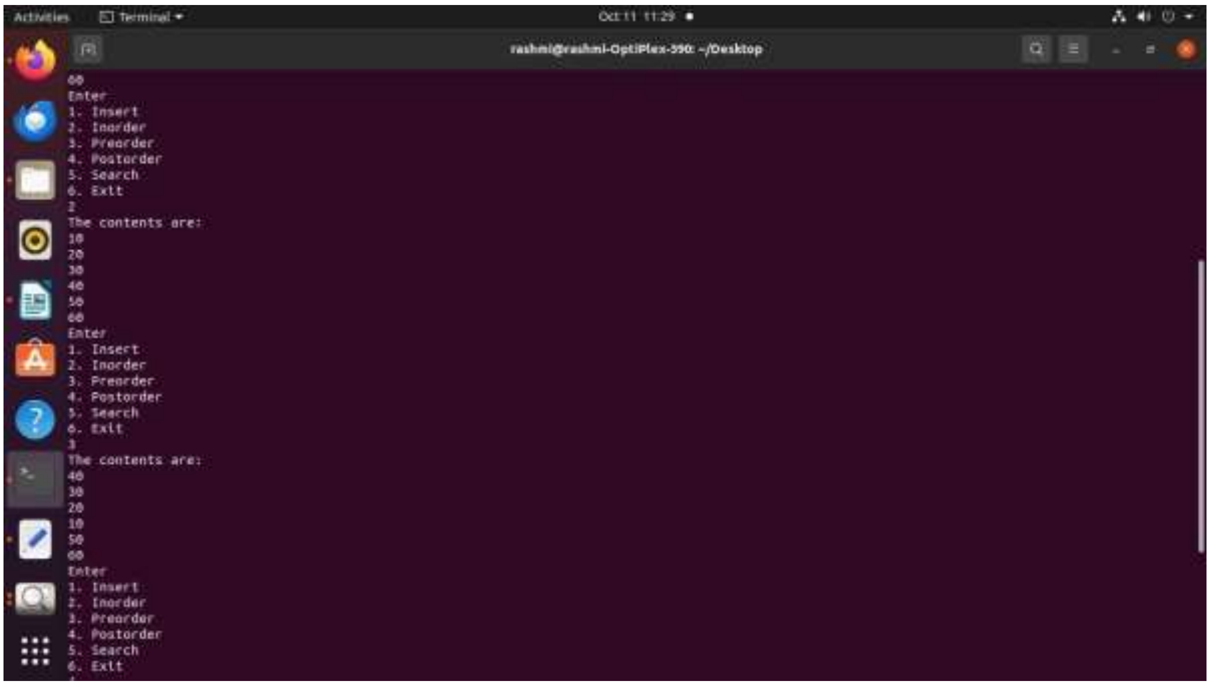
```
                        case 6: exit(0);
                }
        }
}
TNODE * getnode()
{
        TNODE *temp= (TNODE*)malloc(sizeof(TNODE));
        if(temp==NULL)
        {
                printf("Out of memory!!!\n");
                return NULL;
        }
        return temp;
}
TNODE * insert(TNODE * root)
{
   int n,ele,i,flag;
   TNODE *temp,*prev;
   printf("Enter number of nodes\n");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {  printf("Enter element to be inserted:\n");

      scanf("%d", &ele);
      TNODE *newN= getnode();
      newN->data= ele;
      newN->rlink= newN->llink= NULL;

      if(root==NULL)
          {      root=newN; continue; }

          prev=root; temp = root; flag=0;
          while(temp!=NULL)
          {
            prev=temp;
            if (ele==temp->data)
            {
                printf("Redundant data\n");
                flag=1;
                break;
            }
            if(ele<temp->data)
                  temp = temp->llink;
                else
                  temp = temp->rlink;
          }
          if (flag==1) continue;
          if (ele < prev->data)
```

```
                prev->llink = newN;
                else
                  prev->rlink = newN;
        } // end of for
              return root;
    }
    void inorder(TNODE * root)
    {
            if(root!=NULL)
            {
                    inorder(root->llink);
                    printf("%d\n", root->data);
                    inorder(root->rlink);
            }
    }
    void preorder(TNODE * root)
    {
            if(root!=NULL)
            {
                    printf("%d\n", root->data);

                    preorder(root->llink);
                    preorder(root->rlink);
            }
    }
    void postorder(TNODE * root)
    {
            if(root!=NULL)
            {
                    postorder(root->llink);
                    postorder(root->rlink);
                    printf("%d\n", root->data);
            }
    }
    int search( TNODE * root, int key)
    {
            if(root!=NULL)
            {
                    if(root->data==key)
                            return key;
                    if(key < root->data)
                            return search(root->llink, key);
                    return search(root->rlink, key);
            }
            return -1;
    }
```

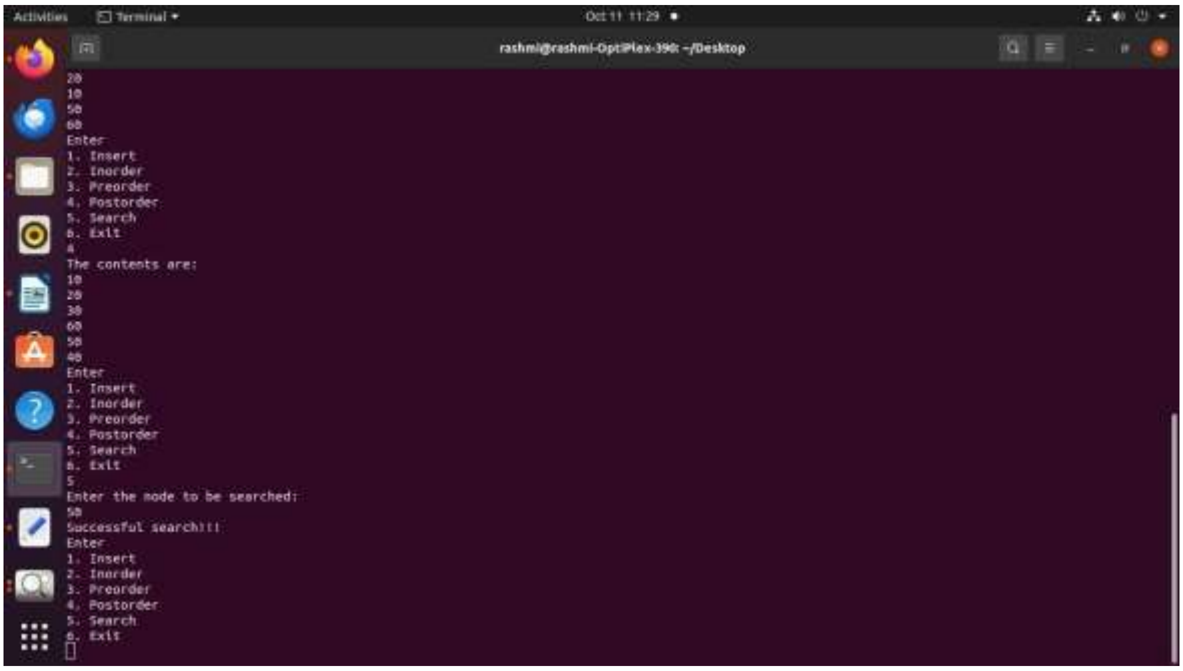
Output:



```
1. Insert
2. Inorder
3. Preorder
4. Postorder
5. Search
6. Exit
1
Enter number of nodes
6
Enter element to be inserted:
40
Enter element to be inserted:
30
Enter element to be inserted:
20
Enter element to be inserted:
10
Enter element to be inserted:
50
Enter element to be inserted:
60
Enter
1. Insert
2. Inorder
3. Preorder
4. Postorder
5. Search
6. Exit
2
The contents are:
10
20
30
40
50
60
Enter
1. Insert
```



```
60
Enter
1. Insert
2. Inorder
3. Preorder
4. Postorder
5. Search
6. Exit
2
The contents are:
10
20
30
40
50
60
Enter
1. Insert
2. Inorder
3. Preorder
4. Postorder
5. Search
6. Exit
3
The contents are:
40
30
20
10
50
60
Enter
1. Insert
2. Inorder
3. Preorder
4. Postorder
5. Search
6. Exit
```



```
20
10
50
60
Enter
1. Insert
2. Inorder
3. Preorder
4. Postorder
5. Search
6. Exit
4
The contents are:
10
20
30
60
50
40
Enter
1. Insert
2. Inorder
3. Preorder
4. Postorder
5. Search
6. Exit
5
Enter the node to be searched:
50
Successful search!!!
Enter
1. Insert
2. Inorder
3. Preorder
4. Postorder
5. Search
6. Exit
```

Program 11

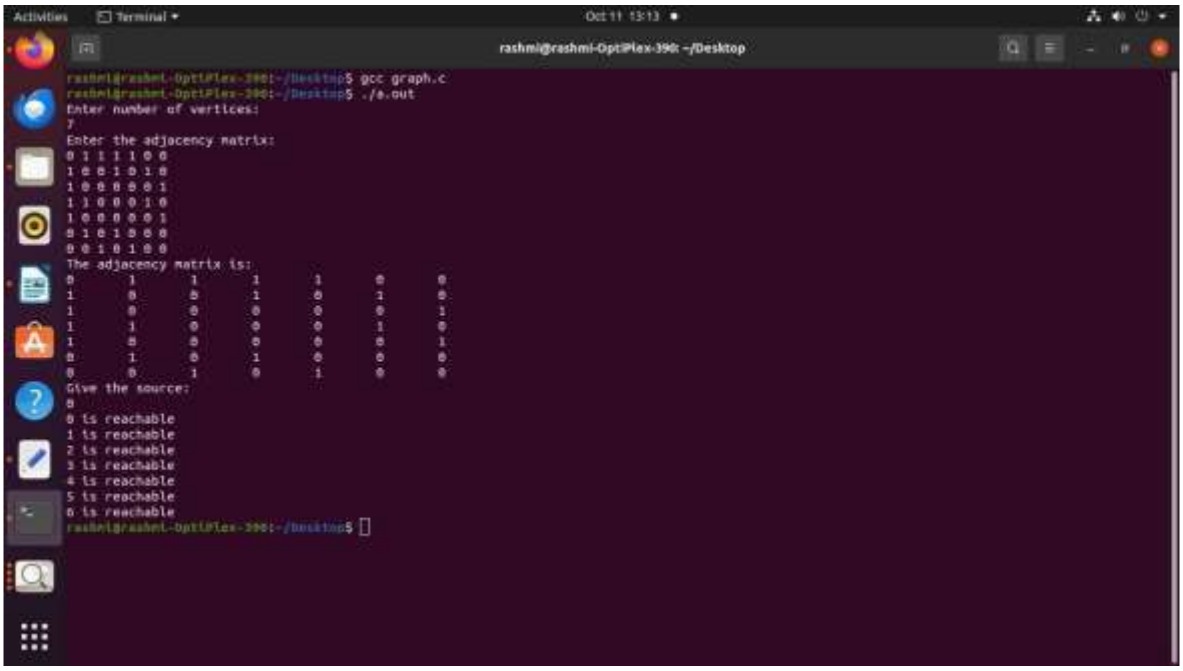
Develop a Program in C for the following operations on Graph(G) of Cities

- **a. Create a Graph of N cities using Adjacency Matrix.**
- **b. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method**

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 20
void bfs(int, int, int [][SIZE], int []);
int main() {
        int n, amat[SIZE][SIZE], source, visited[SIZE], i, j;
        printf("Enter number of vertices:\n");        scanf("%d", &n);
        printf("Enter the adjacency matrix:\n");
        for(i=0; i<n; i++)
         for(j=0; j<n; j++)
          scanf("%d", &amat[i][j]);
        printf("The adjacency matrix is:\n");
        for(i=0; i<n; i++) {
                for(j=0; j<n; j++)
                        printf("%d\t", amat[i][j]);
                printf("\n");
        }
        printf("Give the source:\n");
        scanf("%d", &source);
        for(i=0; i<n; i++)      visited[i]=0;
        bfs(n, source, amat, visited);
        for(i=0;i<n;i++)
          if(visited[i]==0)
            printf("%d is not reachable\n", i);
          else
            printf("%d is reachable\n",i);
        return 0;
}
void bfs(int n, int source, int amat[][SIZE], int visited[]) {
        int q[SIZE], r=0, f=0, u, v;
        visited[source]=1;      q[r]=source;
        while(f<=r)
        {
                u= q[f++];
                for(v=0;v<n;v++)
                {
                        if((amat[u][v]==1)&(visited[v]==0))
                        {
```

```
                                        q[++r]=v;
                                        visited[v]=1;
                                }
                        }
                } }
```

Output:



```
rashmi@rashmi-OptiPlex-390:~/Desktop$ gcc graph.c
rashmi@rashmi-OptiPlex-390:~/Desktop$ ./a.out
Enter number of vertices:
7
Enter the adjacency matrix:
0 1 1 1 1 0 0
1 0 0 1 0 1 0
1 0 0 0 0 0 1
1 1 0 0 0 1 0
1 0 0 0 0 0 1
0 1 0 1 0 0 0
0 0 1 0 1 0 0
The adjacency matrix is:
0       1       1       1       1       0       0
1       0       0       1       0       1       0
1       0       0       0       0       0       1
1       1       0       0       0       1       0
1       0       0       0       0       0       1
0       1       0       1       0       0       0
0       0       1       0       1       0       0
Give the source:
0
0 is reachable
1 is reachable
2 is reachable
3 is reachable
4 is reachable
5 is reachable
6 is reachable
rashmi@rashmi-OptiPlex-390:~/Desktop$ 
```

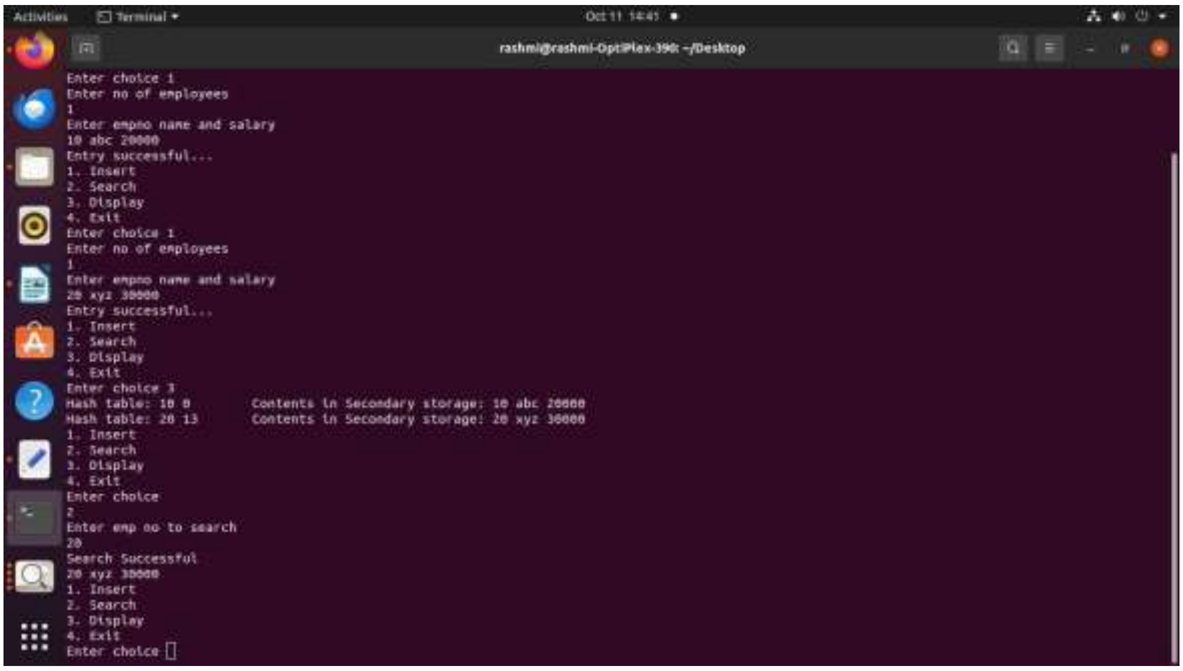
Program 12

Given a File of N employee records with a set K of Keys (4-digit) which uniquely determine
the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Develop a Program in C that uses Hash function H: K →L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.

```
#include  <stdio.h>
#include <stdlib.h>
#define HZ 3
FILE  *fp;
struct employee
{
  int empno;
  char name[20];
  int sal;
};
typedef struct employee EMP;

struct hashtable
{
  int key; // emp no; unique key value
  long int addr; //OFFSET address in file
};
typedef struct hashtable ht;

int hashval(int num) {
  int key;
  key = num % HZ; // division method, being used to calculate hash value
  return key;
}

void search(FILE *fp,ht *h, int n)
{
  EMP a; int hindex,countindex;
  printf("Enter emp no to search\n");
  scanf("%d",&(a.empno));
  hindex = hashval(a.empno);
  countindex=hindex;
  while (h[hindex].key != a.empno)
  {
        hindex = (hindex + 1 ) % HZ;
```

```
            if (countindex == hindex)
            {
            printf("Search unsuccessful\n");
            return;
            }
    }
    printf("Search Successful\n");
    fseek(fp,h[hindex].addr,SEEK_SET);
    //fread(&a,sizeof(EMP),1,fp); // stores information in binary format
    fscanf(fp,"%d%s%d",&(a.empno),a.name,&(a.sal));
    printf("%d %s %d\n",a.empno,a.name,a.sal);
}

void display(FILE *fp, ht *h,int n)
{
    EMP a; int j,i;
    for(i=0;i<HZ;i++)
    {
            if (h[i].key != -1)
            {
            printf("Hash table: %d %d |\t",h[i].key,h[i].addr);
            fseek(fp, h[i].addr,SEEK_SET);
            //fread(&a,sizeof(EMP),1,fp);
            fscanf(fp,"%d%s%d",&(a.empno),a.name,&(a.sal));
    printf("Contents in Secondary storage: %d %s %d\n",a.empno,a.name,a.sal);
            }
    }
}

void insert(FILE *fp,ht *h,int n)
{
    EMP a; int flag=0;
    int i,hindex,countindex;
    for(i=0; i<n; i++,flag=0)
    {
            printf("Enter empno name and salary\n");
            scanf("%d%s%d",&(a.empno),a.name,&(a.sal));
            hindex = hashval(a.empno); // hash value is calculated
            countindex=hindex;
            while(h[hindex].key != -1)
            //checking whether hindex position in h is empty
            {
            hindex = (hindex+1) % HZ; flag=1;
            if (hindex == countindex)
            {
            printf("Entry not possible... \n");
            return;
            }
            }
            h[hindex].key = a.empno;
            fseek(fp,0,SEEK_END);
            h[hindex].addr = ftell(fp);
            //fwrite(&a,sizeof(EMP),1,fp);
            fprintf(fp,"%d %s %d\n",a.empno, a.name, a.sal);
            printf("Entry successful...\n");
            if (flag) printf("Linear probing used\n");
  }
}

int main() {
  ht h[HZ]; EMP d;
  int n,ch;
  fp = fopen("emp.txt","w+");
  for(n=0;n<HZ;n++) //assigning key to -1
            h[n].key = -1;
  for(;;)
  {
            printf("1. Insert\n2. Search\n3. Display\n4. Exit\nEnter choice ");
            scanf("%d",&ch);
            //rewind(fp);
            switch(ch)
            {
            case 1: printf("Enter no of employees\n"); scanf("%d",&n);
            insert(fp,h,n);  break;
            case  2:  search(fp,h,n);  break;
            case 3: display(fp,h,n); break;
            case 4: exit(0);
            }
  }
  return 0;
}
```

Output:



```
Enter choice 1
Enter no of employees
1
Enter empno name and salary
10 abc 20000
Entry successful...
1. Insert
2. Search
3. Display
4. Exit
Enter choice 1
Enter no of employees
1
Enter empno name and salary
20 xyz 30000
Entry successful...
1. Insert
2. Search
3. Display
4. Exit
Enter choice 3
Hash table: 10 0        Contents in Secondary storage: 10 abc 20000
Hash table: 20 13       Contents in Secondary storage: 20 xyz 30000
1. Insert
2. Search
3. Display
4. Exit
Enter choice
2
Enter emp no to search
20
Search Successful
20 xyz 30000
1. Insert
2. Search
3. Display
4. Exit
Enter choice
```

Appendix A

This section comprises of additional programs from each module.

Write a C program for the following:

Module 1

1. Accept and print the employee details using structure pointers and structure variable.
2. Print all Distinct (Unique) Elements in the given Array.
3. Demonstrate the size of pointers for each data type.
4. Demonstrate the difference between pointer to an integer and pointer to an array.
5. a) pass array as parameter to the function. b) pass pointer as parameter to the function c) return type of the function should be a pointer d) pass structure variable as parameter to the function e) return type of the function should be a structure variable
6. a) pass structure pointer as parameter to the function b) return type of the function should be a structure pointer.
7. Find K most occurring elements in the given Array.
8. How to Reverse a String using Stack
9. How to Reverse a Stack using Recursion
10. Check for Balanced Brackets in an expression (well-formedness)

Module 2

11. Find the middle of a given linked list.
12. Count nodes in Circular linked list
13. Convert singly linked list into circular linked list.
14. Remove duplicates from a sorted linked list.
15. An interesting method to print the reverse of a linked list.

Module 3

16. Program to find size of Doubly Linked List
17. Merge Sort for Doubly Linked List
18. Sparse Matrix and its representations
19. Get level of a node in a Binary Tree
20. Program to Determine if the given two trees are identical or not.

Module-4

21. Print Right View of a Binary Tree
22. A program to check if a Binary Tree is BST or not.
23. Search for an element in BST.
24. Create a binary tree using pointers.
25. Adjacency matrix representation of undirected graph.

Module 5

26. Find whether an array is subset of another array.
27. Create a file for reading and writing.
28. Create a priority queue using binary max heap.
29. Create a priority queue using arrays and linked list.
30. Recursive implementation of optimal Binary Search Tree.

Appendix B

Viva Questions

This section comprises of possible viva questions that can be asked in examination.

Module 1

1. Consider a two-dimensional array A [20][10]. Assume 4 words per memory cell, the base address of array A is 100, elements are stored in row-major order and first element is A [0][0]. What is the address of A [11][5]?
2. How can we describe an array in the best possible way?
3. What is the size of int arr [9] assuming that int is of 4 bytes?
4. When the user tries to delete the element from the empty stack then the condition is said to be a ____________
5. If the size of the stack is 10 and we try to add the 11th element in the stack, then the condition is known as_______
6. Which data structure is mainly used for implementing the recursive algorithm?
7. If the elements '1', '2', '3' and '4' are added in a stack, what would be the order for the removal?
8. What is (void*)0?
9. If a variable is a pointer to a structure, then which operator is used to access data members of the structure through the pointer variable?
10. The operator used to get value at address stored in a pointer variable is ____________________

Module 2

11. Explain the applications of stack.
12. Define linked list.
13. Convert the expression (A + B) * C - (D - E) ^ (F + G)) to equivalent Prefix and Postfix notations.
14. Minimum number of queues needed to implement the priority queue?
15. Which one of the following nodes is considered the top of the stack if the stack is implemented using the linked list?
16. The minimum number of stacks required to implement a queue is?
17. What is the outcome of the prefix expression +, -, *, 3, 2, /, 8, 4, 1?
18. If the elements '1', '2', '3' and '4' are inserted in a queue, what would be the order for the removal?

19. What is the overflow condition if linear queue is implemented using an array with a size MAX_SIZE?
20. Explain the difference between stack and queue.

Module 3:

21. In the linked list implementation of queue, where will the new element be inserted?

22. Which data structure is the best for implementing a priority queue?

23. In the Deque implementation using singly linked list, what would be the time complexity of deleting an element from the rear end?

24. If circular queue is implemented using array having size MAX_SIZE in which array index starts with 0, front points to the first element in the queue, and rear points to the last element in the queue. Which condition is used to specify that the circular queue is empty?

25. Explain the concept of doubly linked list.

26. Explain dequeue.

27. What is NODE.

28. Differentiate singly linked list and doubly linked list.

29. Define Threaded Binary Tree.

30. Explain Inorder, Preorder and Postorder.

Module 4:

31. What is the maximum number of children that a node can have in a binary tree?

32. Differentiate binary tree and binary search tree.

33. What are selection trees.

34. Define graph. Explain its importance in data structure.

35. What do you mean by abstract data type.

36. Explain the concept of Disjoint sets. Mention its application.

37. What are counting binary trees.

38. Mention the applications of BST.

39. Define forest.

40. Differentiate BFS and DFS

Module 5:

41. Define Hashing

42. If h is chosen from a universal collection of hash functions and is used to hash n keys into a table of size m, where n ≤ m, the expected number of collisions involving a particular key x is less than________.

43. A dictionary has a set of   and each key has a single associated value.

44. Consider a hash table of size seven, with starting index zero, and a hash function (3x + 4) mod7. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using closed hashing? Note that '_' denotes an empty location in the table.

45. Which hashing technique is free from clustering?

46. Which hash function is used in the division method?

47. When it would be optimal to prefer Red-black trees over AVL trees?

48. What is the disadvantage of using splay trees?

49. When it would be optimal to prefer Red-black trees over AVL trees?

50. If several elements are competing for the same bucket in the hash table, what is it called?