|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INDEX** | | | | |
| **Sl no** | **DATE** | ***TITLE OF THE PROGRAM*** | **PAGE NO** | **sign** |
| 1 | 20/11/2023 | Develop a Program in C for the following:   1. 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). 2. 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 | 27/11/2023 | Develop a Program in C for the following operations on Strings.   1. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP). 2. 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 | 04/12/2023 | Develop a menu driven Program in C for the following operations  on STACK of Integers.  (Array Implementation of Stack with maximum size MAX)   1. Push an Element on to Stack 2. Pop an Element from Stack 3. Demonstrate how Stack can be used to check Palindrome 4. Demonstrate Overflow and Underflow situations on Stack 5. Display the status of Stack 6. Exit   Support the program with appropriate functions for each of the above operations. |  |  |
| 4 | 11/12/2023 | 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 | 18/12/2023 | Develop a Program in C for the following Stack Applications   1. Evaluation of Suffix expression with single digit operands and operators: +, -, \*, /, %, ^ . 2. Solving Tower of Hanoi problem with n disks |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 25/12/2023 | Develop a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)   1. Insert an Element on to Circular QUEUE . 2. Delete an Element from Circular QUEUE . 3. Demonstrate Overflow and Underflow situations on Circular QUEUE. 4. Display the status of Circular QUEUE 5. Exit.   Support the program with appropriate functions for each of the above operations. |  |  |
| 7 | 01/01/2024 | Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with thefields: *USN, Name, Programme, Sem, PhNo.*   1. Create a SLL of N Students Data by using *front insertion*. 2. Display the status of SLL and count the number of nodes in it 3. Perform Insertion / Deletion at End of SLL. 4. Perform Insertion / Deletion at Front of SLL(Demonstration of stack). 5. Exit. |  |  |
| 8 | 08/01/2024 | 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 .*   1. Create a DLL of N Employees Data by using *end insertion*. 2. Display the status of DLL and count the number of nodes in it 3. Perform Insertion and Deletion at End of DLL . 4. Perform Insertion and Deletion at Front of DLL. 5. Demonstrate how this DLL can be used as Double Ended Queue). 6. Exit. |  |  |
| 9 | 15/01/2024 | Develop a Program in C for the following operationson Singly Circular Linked List (SCLL) with header nodes   1. Represent and Evaluate a Polynomial P(x,y,z) = 6x 2 y 2 z-4yz 5 +3x 3 yz+2xy 5 z-2xyz 3 2. 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 | 22/01/2024 | Develop a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers .   1. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2 2. Traverse the BST in Inorder, Preorder and Post Order 3. Search the BST for a given element (KEY) and report the appropriate message 4. Exit. |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 | 05/02/2024 | Develop a Program in C for the following operations on Graph(G) of Cities .   1. Create a Graph of N cities using Adjacency Matrix. 2. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method. |  |  |
| 12 | 12/02/2024 | 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 inear probing. |  |  |

**Lab 1 :**

Develop a Program in C for the following:

1. 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).
2. 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<stdlib.h>

#include<string.h>

#include<conio.h>

typedef struct

{

char \*name;

int date;

char \*description;

}Day;

void create(Day \*week, int size)

{

for (int i = 0; i < size; i++)

{

week[i].name = (char \*)malloc(20 \* sizeof(char));

week[i].description = (char \*)malloc(100 \* sizeof(char));

}

}

void read(Day\*week,int size)

{

for (int i = 0; i < size; i++)

{

printf("Enter name of day %d:", i+1);

scanf("%s", week[i].name);

printf("Enter the date:");

scanf("%d", &week[i].date);

printf("Enter description");

scanf(" %[^\n]", week[i].description);

}

}

void display(Day \*week, int size )

{

printf("\n Week's Activity Details:\n");

for(int i = 0; i < size; i++)

{

printf("Day %s, Date:%d, Activity:%s\n",week[i].name,week[i].date,week[i].description);

}

}

int main()

{

int size = 7;

Day\*week = (Day \*)malloc(size \* sizeof(Day));

create(week, size);

read(week , size);

display(week, size);

for (int i = 0;i < size; i++)

{

free(week[i].name);

free(week[i].description);

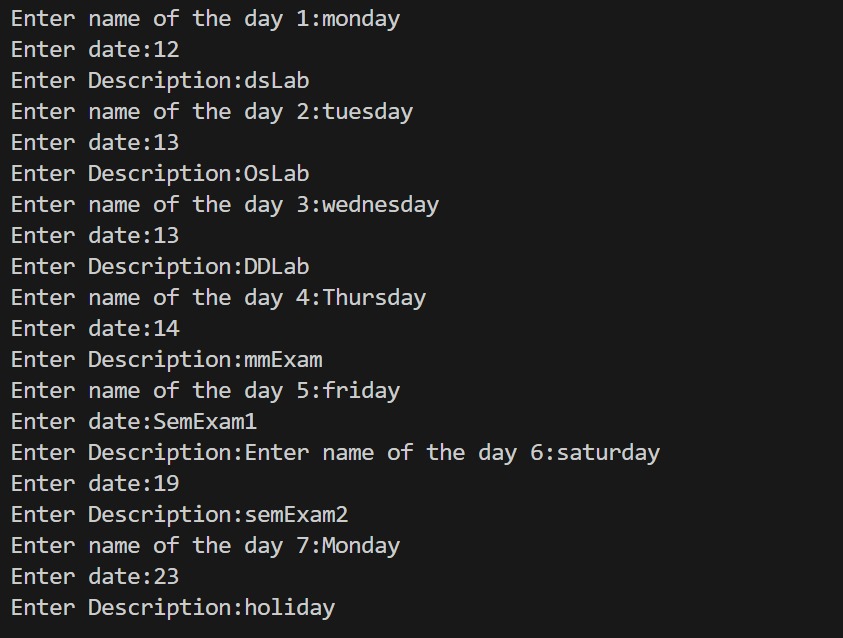
}

free(week);

getch();

return 0;

}



**Lab : 2**

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

1. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP).
2. 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>

void main()

{

char STR[100],PAT[100],REP[100],ans[100];

int i,j,c,m,k,flag=0;

printf("\n Enter the MAIN string:\n");gets(STR);

printf("\n Enter the PATTERN string:\n");gets(PAT);

printf("\n Enter the REPLACE string:\n");gets(REP);

i=m=c=j=0;while(STR[c]!='\0'){

if (STR[m]==PAT[i]

){i++;m++;

flag=1;

if (PAT[i]=='\0')

{

for (k=0;REP[k]!='\0';k++,j++)

ans[j]=REP[k];

i=0;

c=m;

}

}

else

{

ans[j]=STR[c];

j++;c++;

m=c;i=0;

}

}

if (flag==0)

{

printf("Pattern doesnt found!!!");

}

else

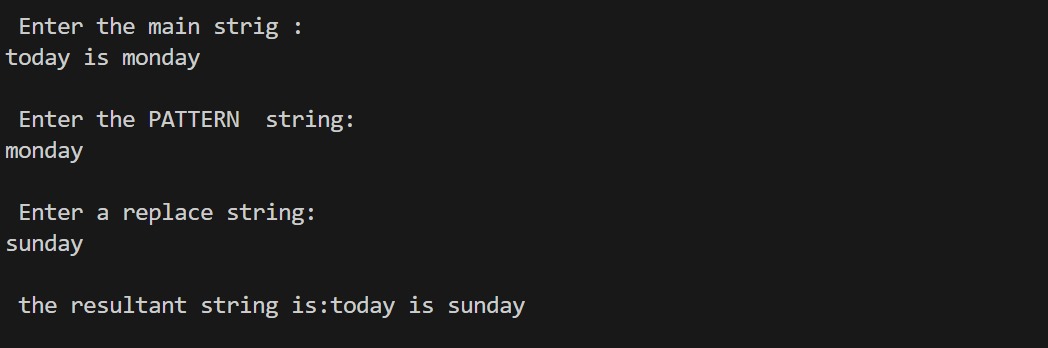
{

ans[j]='\0';

printf("\n The RESULTANT string is :%s\n",ans);

}

}



**Lab : 3**

Develop a menu driven Program in C for the following operations

on STACK of Integers.

(Array Implementation of Stack with maximum size MAX)

1. Push an Element on to Stack
2. Pop an Element from Stack
3. Demonstrate how Stack can be used to check Palindrome
4. Demonstrate Overflow and Underflow situations on Stack
5. Display the status of Stack
6. Exit

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

#include<stdio.h>

#include<conio.h>

#include<process.h>

void main()

{

int ch,i,j,stack[5],sp=0;

int r,t,n,rev;

clrscr();

do

{

printf("\n STACK ");

printf("\n 1. Push");

printf("\n 2. Pop");

printf("\n 3. Display");

printf("\n 4. palindrome");

printf("\n 0 Exit");

printf("\n Enter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:

if(sp>4)

{

printf("\n the stack is full:");

}

else

{

printf("\n Enter the number to push:");

scanf("%d",&stack[sp]);

sp=sp+1;

}

break;

case 2:

if(sp<=0)

{

printf("\n The Stack is empty:");

getch();

}

else

{

sp=sp-1;

printf("\n The poped element is=%d",stack[sp]);

}

break;

case 3:

if(sp<=0)

{

printf("\n Stack is Empty");

getch();

}

else

{

printf("\n Stack elements are\n");

for(i=sp-1;i>=0;i--)

printf("\n%d",stack[i]);

}

break;

case 4:

rev=0;

printf("\n enter your number:");

scanf("%d",&n);

t=n;

do

{

r=n%10;

rev=rev\*10+r;

n=n/10;

}while(n!=0);

if(rev==t)

{

printf("%d is palindrome\n",t);

}

else

printf("%d is not palindrome\n",t);

/\*printf("\nEnter the Number to palindreome:");

scanf("%d",&n);

printf("\n Enter the number to push:");

scanf("%d",&stack[sp]);

sp=sp+1;

char n[20]

if(sp!=0)

{

j=5;i=0;

if(stack[i]==stack[j-1])

if(stack[i+1]==stack[j-2])

if(stack[i+3]==stack[j-3])

printf("\nThe stack is palindrome");

}

else

printf("\nStack is Empty push some elements first");\*/

break;

case 0:

printf("\n stack is terminated");

getch();

exit(0);

break;

default:

printf("\n Invalid choise");

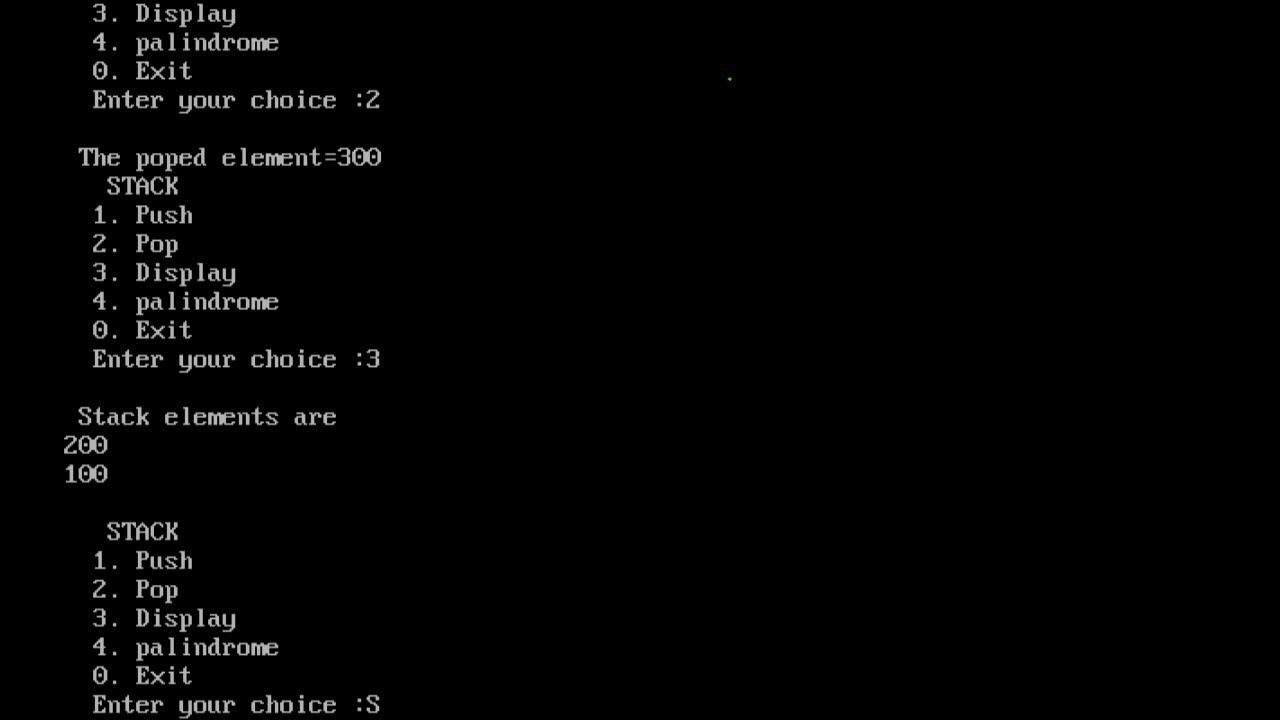
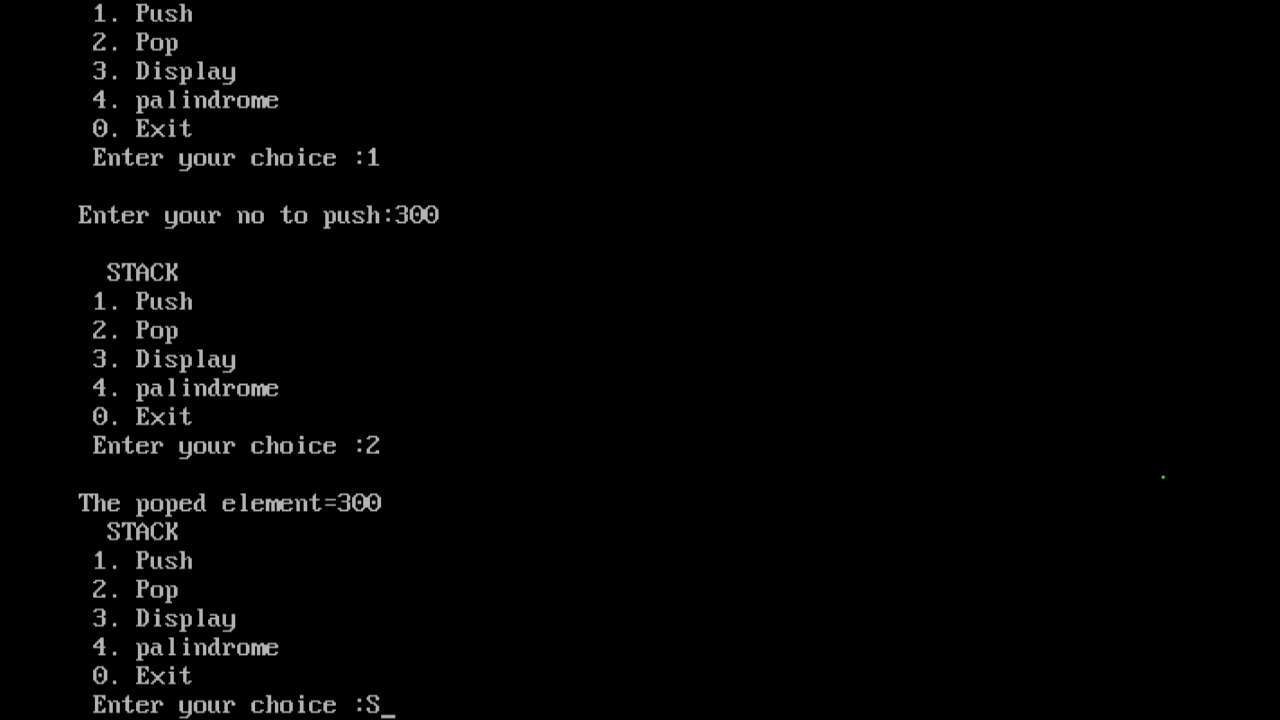
break;

}

}while(ch!=0);

getch();

}



****

**Lab 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<conio.h>

#include<string.h>

int F(char symbol)

{

switch(symbol)

{

case '+':

case '-': return 2;

case '\*':

case '%':

case '/': return 4;

case '^':

case '$': return 5;

case '(': return 0;

case '#': return -1;

default: return 8;

}

}

int G(char symbol)

{

switch(symbol)

{

case '+':

case '-': return 1;

case '\*':

case '%':

case '/': return 3;

case '^':

case '$': return 6;

case '(': return 9;

case ')': return 0;

default: return 7;

}

}

void infix\_postfix(char infix[],char postfix[])

{

int top,i,j;

char S[30];

char symbol;

top=-1;

S[++top]='#';

j=0;

for(i=0;i<strlen(infix);i++)

{

symbol=infix[i];

while(F(S[top])>G(symbol))

postfix[j++]=S[top--];

if(F(S[top])!=G(symbol))

S[++top]=symbol;

else top--;

}

while(S[top]!='#')

postfix[j++]=S[top--];

postfix[j]='\0';

}

void main()

{

char infix[20];

char postfix[20];

clrscr();

printf("\n Enter a valid expression: ");

scanf("%s",infix);

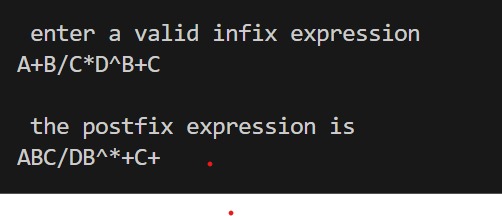
infix\_postfix(infix,postfix);

printf("\n The postfix expression is : ");

printf("%s\n",postfix);

getch();

}



**Lab 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

a)

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define MAX 20

struct stack

{

int top;

float str[MAX];

}s; //stack

char postfix[MAX]; //postfix

void push(float);

float pop();

int isoperand(char);

float operate(float,float,char);

int main()

{

int i=0;

float ans,op1,op2;

clrscr();

printf("Enter expression:");

scanf("%s",postfix);

while(postfix[i]!='\0')

{

if(isoperand(postfix[i]))

push(postfix[i]-48);

else

{

op1=pop();

op2=pop();

ans=operate(op1,op2,postfix[i]);

push(ans);

printf("%f %c %f = %f\n",op2,postfix[i],op1,ans);

}

i++;

}

printf("%f",s.str[s.top]);

getch();

return 0;

}

int isoperand(char x)

{

if(x>='0'&&x<='9')

return 1;

else

return 0;

}

void push(float x)

{

if(s.top==MAX-1)

printf("Stack is full \n STACK OVERFLOW \n");

else

{

s.top ++;

s.str[s.top]=x;

}

}

float pop()

{

if(s.top==-1)

{

printf("Stack is empty \n STACK UNDERFLOW \n");

getch();

}

else

{

s.top--;

return s.str[s.top+1];

}

}

float operate(float op1, float op2, char a)

{

switch(a)

{

case '+': return op2+op1;

case '-': return op2-op1;

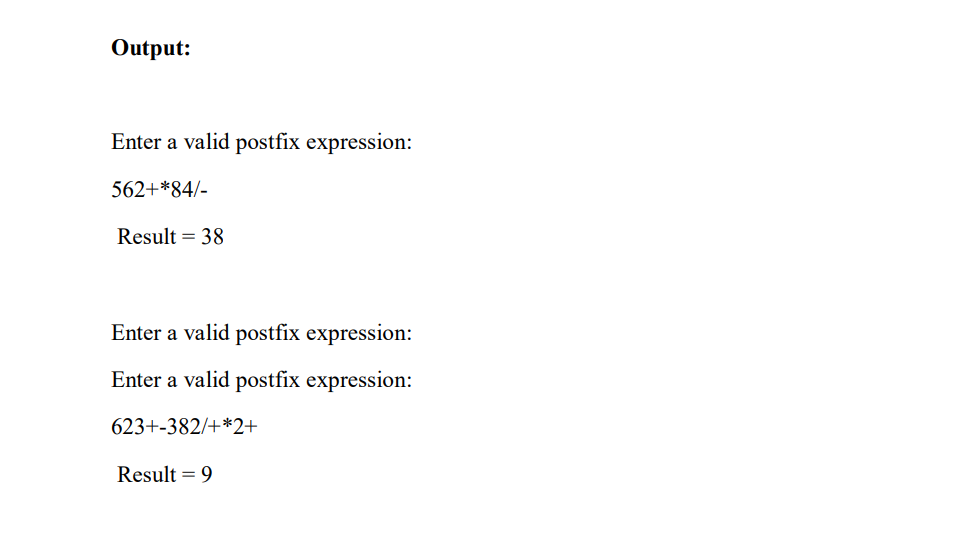
case '\*': return op2\*op1;

case '/': return op2/op1;

case '^': return pow(op2,op1);

}

}



b)

#include<stdio.h>

#include<math.h>

void tower(int n, int source, int temp,int destination)

{

if(n == 0)

return;

tower(n-1, source, destination, temp);

printf("\nMove disc %d from %c to %c", n, source, destination);

tower(n-1, temp, source, destination);

}

void main()

{

int n;

clrscr();

printf("\nEnter the number of discs: ");

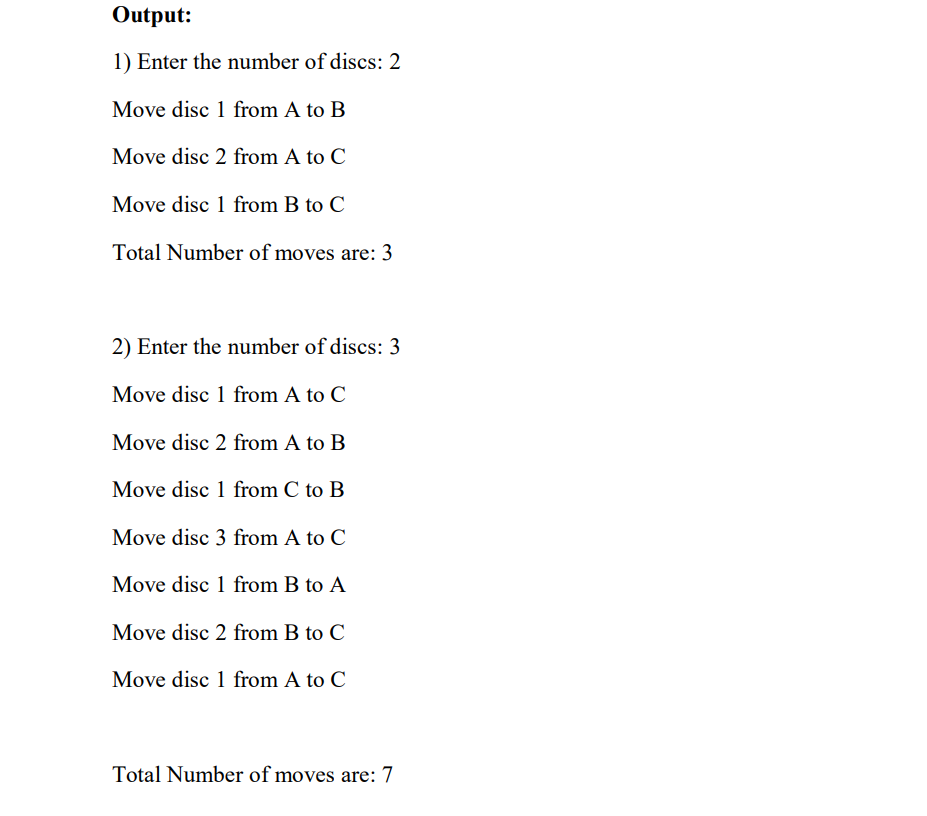
scanf("%d", &n);

tower(n, 'A', 'B', 'C');

printf("\n\nTotal number of moves are: %d", (int)pow(2,n)-1);

getch();

}



**Lab 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)

1. Insert an Element on to Circular QUEUE .
2. Delete an Element from Circular QUEUE .
3. Demonstrate Overflow and Underflow situations on Circular QUEUE.
4. Display the status of Circular QUEUE
5. Exit.

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

#include<stdio.h>

#include<conio.h>

#define SIZE 5

int CQ[SIZE];

int front=-1;

int rear=-1, ch;

int IsCQ\_Full();

int IsCQ\_Empty();

void CQ\_Insert(int);

void CQ\_Delet();

void CQ\_Display();

void main()

{

clrscr();

printf("\n\t\t PROGRAM ON CIRCULAR QUEUE ");

printf("\n 1.Insert \n 2.Delete \n 3.Display \n 4.Exit \n");

while(1)

{

int ele;

printf("\n Enter your choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1:

if(IsCQ\_Full())

printf("\n Circular Queu Overflow\n");

else

{

printf("\n Enter the element to be inserted : ");

scanf("%d",&ele);

CQ\_Insert(ele);

}

break;

case 2:

if(IsCQ\_Empty())

printf("\n Circular Queue Underflow \n");

else

CQ\_Delet();

break;

case 3:

if(IsCQ\_Empty())

printf("\n Circular Queue Underflow \n");

else

CQ\_Display();

break;

case 4:

exit(0);

}

}

}

void CQ\_Insert(int item)

{

if(front==-1)

front++;

rear = (rear+1)%SIZE;

CQ[rear] =item;

}

void CQ\_Delet()

{

int item; item=CQ[front];

printf("\n Deleted element is: %d ",item);

front = (front+1)%SIZE;

}

void CQ\_Display()

{

int i;

if(front==-1)

printf("\n Circular Queue is Empty \n");

else

{

printf("\n Elements of the circular queue are : ");

for(i=front;i!=rear;i=(i+1)%SIZE)

{

printf("%d\t",CQ[i]);

}

printf("%d\n",CQ[i]);

}

}

int IsCQ\_Full()

{

if(front==(rear+1)%SIZE)

return 1;

return 0;

}

int IsCQ\_Empty()

{

if(front == -1)

return 1;

else if(front == rear)

{

printf("\n Deleted element is: %d \n",CQ[front]);

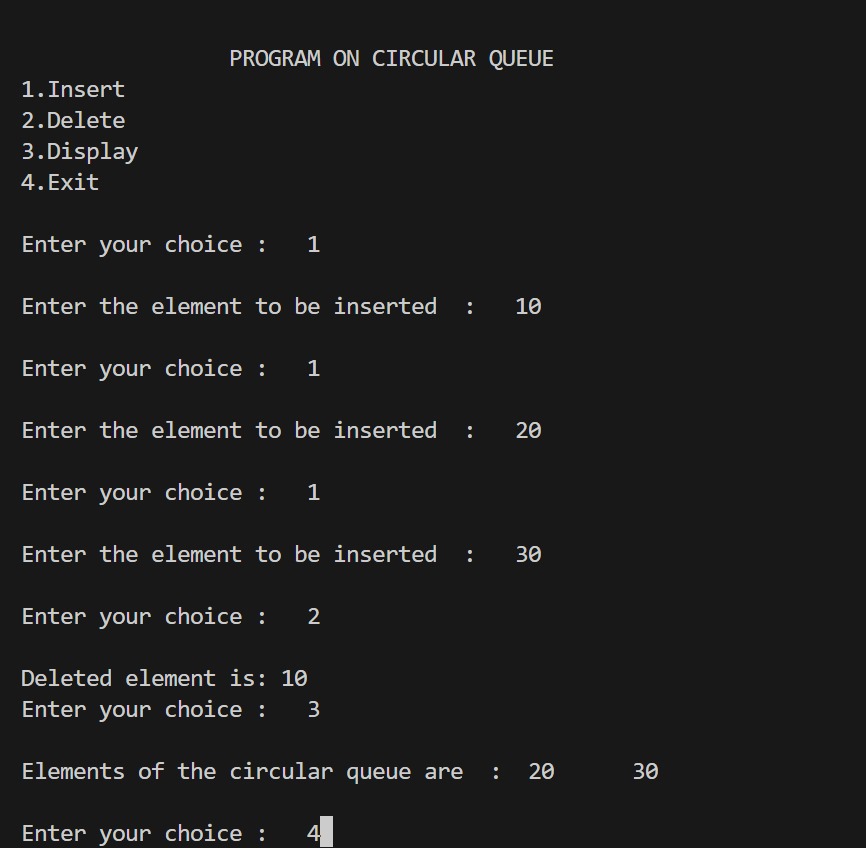
front=-1;

return 1;

}

return 0;

}



**Lab 7 :**

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

1. Create a SLL of N Students Data by using *front insertion*.
2. Display the status of SLL and count the number of nodes in it
3. Perform Insertion / Deletion at End of SLL.
4. Perform Insertion / Deletion at Front of SLL(Demonstration of stack).

Exit.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int count=0;

struct stud

{

long int ph;

int sem;

char name[15],usn[15],brnch[8];

struct stud \*next;

}

\*head=NULL,\*tail=NULL,\*temp=NULL,\*temp1;

void create(long long int n,int s,char na[20],char u[15],char b[5])

{

if(head==NULL)

{

head=(struct stud\*)malloc(1\*sizeof(struct stud));

head->ph=n;

head->sem=s;

strcpy(head->name,na);

strcpy(head->usn,u);

strcpy(head->brnch,b);

head->next=NULL;

tail=head;

count++;

}

else

{

temp=(struct stud\*)malloc(1\*sizeof(struct stud));

temp->ph=n;

temp->sem=s;

strcpy(temp->name,na);

strcpy(temp->usn,u);

strcpy(temp->brnch,b);

temp->next=NULL;

tail->next=temp;

tail=temp;

count++;

}

}

void display()

{

temp1=head;

if(temp1==NULL)

{

printf("\nlist is empty\n");

}

else

{

printf("student details are as follows:\n");

while(temp1!=NULL)

{

printf(" \n");

printf("NAME:%s\nUSN:%s\nBRANCH:%s\nSEM:%d\nPHONE NO.:%lld\n",temp1->name,temp1->usn,temp1->brnch,temp1->sem,temp1->ph);

printf(" \n");

temp1=temp1->next;

}

printf("no. of nodes=%d\n",count);

}

}

void insert\_head(long long int n,int s,char na[15],char u[15],char b[8])

{

temp=(struct stud\*)malloc(1\*sizeof(struct stud));

temp->ph=n;

temp->sem=s;

strcpy(temp->name,na);

strcpy(temp->usn,u);

strcpy(temp->brnch,b);

temp->next=head;

head=temp;

count++;

}

void insert\_tail(long long int n,int s,char na[15],char u[15],char b[8])

{

temp=(struct stud\*)malloc(1\*sizeof(struct stud));

temp->ph=n;

temp->sem=s;

strcpy(temp->name,na);

strcpy(temp->usn,u);

strcpy(temp->brnch,b);

tail->next=temp;

temp->next=NULL;

tail=temp;

count++; }

void delete\_head()

{

temp1=head;

if(temp1==NULL)

{

printf("list is empty\n");

}

else

{

head=head->next;

printf("deleted node is:\n");

printf(" \n");

printf("NAME:%s\nUSN:%s\nBRANCH:%s\nSEM:%d\nPHONE NO.:%lld\n",temp1->name,temp1->usn,temp1->brnch,temp1->sem,temp1->ph);

printf(" \n");

free(temp1);

count--;

}

}

void delete\_tail()

{

temp1=head;

if(temp1==NULL)

{

printf("list is empty\n");

}

while(temp1->next!=tail)

{

temp1=temp1->next;

}

printf("deleted node is:\n");

printf(" \n");

printf("NAME:%s\nUSN:%s\nBRANCH:%s\nSEM:%d\nPHONE NO.:%lld\n",tail->name,tail->usn,tail->brnch,tail->sem,tail->ph); printf(" \n");

free(tail);

tail=temp1;

tail->next=NULL;

count--;

}

void main()

{

int choice;

long long int ph;

int sem;

char name[20],usn[15],brnch[5];

clrscr();

while(1)

{

printf("----------MENU----------\n");

printf("1.create\n2.Insert from head\n3.Insert from tail\n4.Delete from head\n5.Delete fromtail\n6.display\n7.exit\n");

printf("----------------------");

printf("\n Enter your choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter the name usn branch sem phno. of the student respectively\n");

scanf("%s%s%s%d%lld",name,usn,brnch,&sem,&ph);

create(ph,sem,name,usn,brnch);

break;

case 2:

printf("enter the name usn branch sem phno. of the student respectively\n");

scanf("%s%s%s%d%lld",name,usn,brnch,&sem,&ph);

insert\_head(ph,sem,name,usn,brnch);

break;

case 3:

printf("\n Enter the name usn branch sem phno. of the student respectively\n");

scanf("%s%s%s%d%lld",name,usn,brnch,&sem,&ph);

insert\_tail(ph,sem,name,usn,brnch);

break;

case 4:

delete\_head();

break;

case 5:

delete\_tail();

break;

case 6:

display();

break;

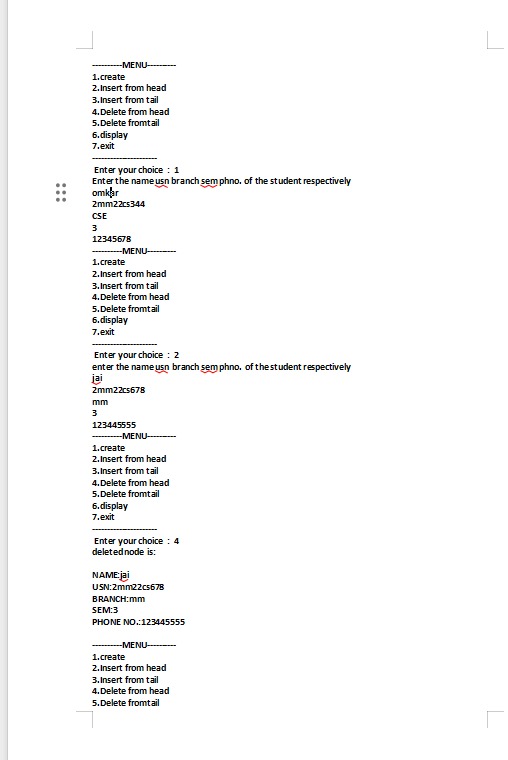
case 7:

exit(0);

default:printf("invalid option\n");

}

}



**Lab 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 .*

1. Create a DLL of N Employees Data by using *end insertion*.
2. Display the status of DLL and count the number of nodes in it
3. Perform Insertion and Deletion at End of DLL .
4. Perform Insertion and Deletion at Front of DLL.
5. Demonstrate how this DLL can be used as Double Ended Queue).
6. Exit.

/\* PROGRAM ON DOUBLE LINKED LIST \*/

/\* TURMWORK NO 8 \*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Enode

{

char ssn[15];

char name[20];

char dept[5];

char designation[10];

int salary;

long long int phno;

struct Enode \*left;

struct Enode \*right;

}\*head=NULL;

struct Enode \*tail,\*temp1,\*temp2;

void create(char [],char [],char [],char [],int ,long long int);

void ins\_beg(char [],char [],char [],char [],int ,long long int);

void ins\_end(char [],char [],char [],char [],int ,long long int);

void del\_beg();

void del\_end();

void display();

int count=0;

void main()

{

int choice;

char s[15],n[20],dpt[5],des[10];

int sal;

long long int p;

while(1)

{

clrscr();

printf("1.Create\n 2.Display\n 3.Insert at beginning\n 4.Insert at End\n 5.Delete at beginning\n 6.Delete at End\n 7.Exit\n");

printf("\n Enter your choice \n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\n Enter the required data(Emp no,Name,Dept,Desig,sal,phone \n");

scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);

create(s,n,dpt,des,sal,p);

break;

case 2:

display();

break;

case 3:

printf("\n Enter the required data (Emp no,Name,Dept,Desig,sal,phone \n");

scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);

ins\_beg(s,n,dpt,des,sal,p);

break;

case 4:

printf("\n Enter the required data(Emp no,Name,Dept,Desig,sal,phone \n");

scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);

ins\_end(s,n,dpt,des,sal,p);

break;

case 5:

del\_beg();

break;

case 6:

del\_end();

break;

case 7:

exit(0);

}

}

}

void create(char s[15],char n[20],char dpt[5],char des[10],int sal,long long int p)

{

if(head==NULL)

{

head=(struct Enode \*)malloc(1\*sizeof(struct Enode));

strcpy(head->ssn,s);

strcpy(head->name,n);

strcpy(head->dept,dpt);

strcpy(head->designation,des);

head->salary=sal;

head->phno=p;

head->left=NULL;

head->right=NULL;

tail=head;

}

else

{

temp1=(struct Enode \*)malloc(1\*sizeof(struct Enode));

strcpy(temp1->ssn,s);

strcpy(temp1->name,n);

strcpy(temp1->dept,dpt);

strcpy(temp1->designation,des);

temp1->salary=sal;

temp1->phno=p;

tail->right=temp1;

temp1->right=NULL;

temp1->left=tail;

tail=temp1;

}

}

void display()

{

temp1=head;

printf("\n Employee Details \n");

while(temp1!=NULL)

{

printf(" \n");

printf("%s\n%s\n%s\n%s\n%d\n%lld\n",temp1->ssn,temp1->name,temp1->dept,temp1->designation,temp1->salary,temp1->phno); printf(" ");

temp1=temp1->right;

}

}

void ins\_beg(char s[15],char n[20],char dpt[5],char des[10],int sal,long long int p)

{

temp1=(struct Enode \* )malloc(1\*sizeof(struct Enode));

strcpy(temp1->ssn,s);

strcpy(temp1->name,n);

strcpy(temp1->dept,dpt);

strcpy(temp1->designation,des);

temp1->salary=sal;

temp1->phno=p;

temp1->right=head;

head->left=temp1;

head=temp1;

temp1->left=NULL;

}

void ins\_end(char s[15],char n[20],char dpt[5],char des[10],int sal,long long int p)

{

temp1=(struct Enode \*)malloc(1\*sizeof(struct Enode));

strcpy(temp1->ssn,s);

strcpy(temp1->name,n);

strcpy(temp1->dept,dpt);

strcpy(temp1->designation,des);

temp1->salary=sal;

temp1->phno=p;

tail->right=temp1;

temp1->left=tail;

temp1->right=NULL;

tail=temp1;

}

void del\_beg()

{

temp1=head->right;

free(head);

head=temp1;

head->left=NULL;

}

void del\_end()

{

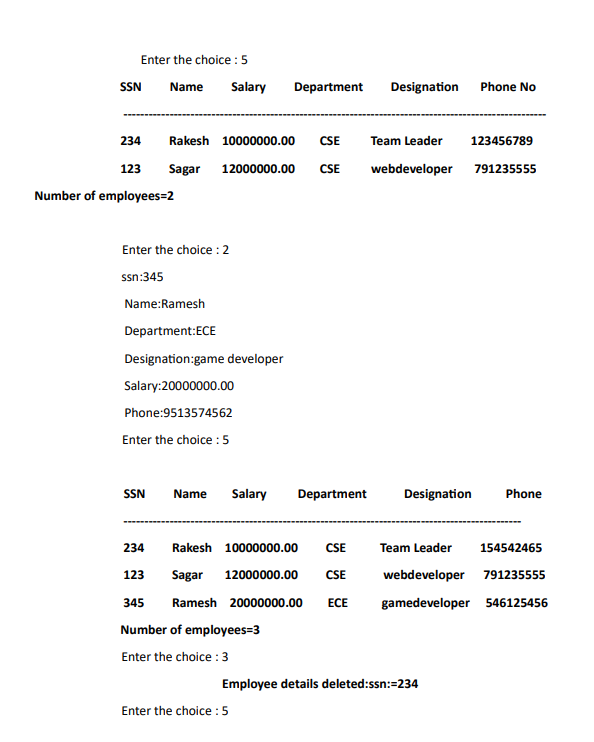
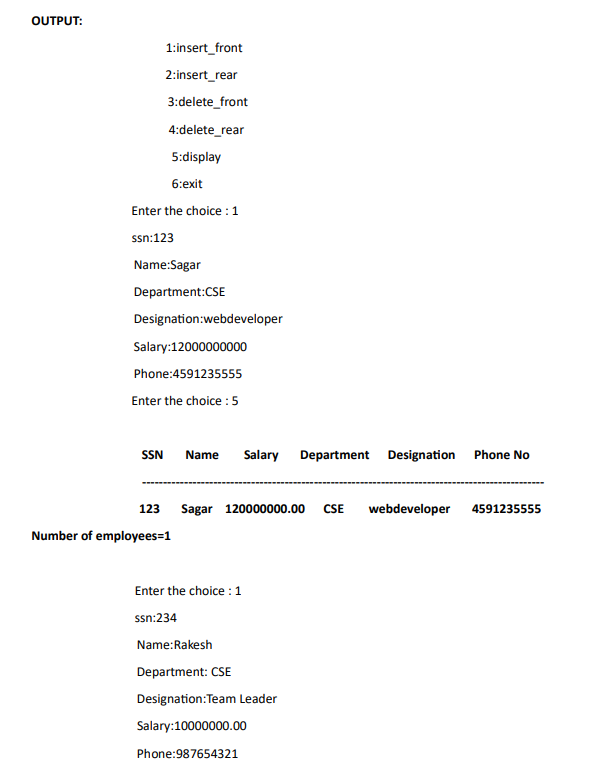
temp1=tail->left;

free(tail);

tail=temp1;

tail->right=NULL;

}



**Lab 9 :**

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

1. Represent and Evaluate a Polynomial P(x,y,z) = 6x 2 y 2 z-4yz 5 +3x 3 yz+2xy 5 z-2xyz 3
2. 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

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<math.h>

typedef struct node

{

int expo,coef;

struct node \*next;

}node;

node \* insert(node \*,int,int);

node \* create();

node \* add(node \*p1,node \*p2);

int eval(node \*p1);

void display(node \*head);

node \*insert(node\*head,int expo1,int coef1)

{

node \*p,\*q;

p=(node \*)malloc(sizeof(node));

p->expo=expo1;

p->coef=coef1;

p->next=NULL;

if(head==NULL)

{

head=p;

head->next=head;

return(head);

}

if(expo1>head->expo)

{

p->next=head->next;

head->next=p;

head=p;

return(head);

}

if(expo1==head->expo)

{

head->coef=head->coef+coef1;

return(head);

}

q=head;

while(q->next!=head&&expo1>=q->next->expo)

q=q->next;

if(p->expo==q->expo)

q->coef=q->coef+coef1;

else

{

p->next=q->next;

q->next=p;

}

return(head);

}

node \*create()

{

int n,i,expo1,coef1;

node \*head=NULL;

printf("\n\n Enter no of terms of polynomial==>");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("\n Enter coef & expo==> ");

scanf("%d%d",&coef1,&expo1);

head=insert(head,expo1,coef1);

}

return(head);

}

node \*add(node \*p1,node \*p2)

{

node \*p;

node \*head=NULL;

printf("\n\n\n Addition of polynomial==> ");

p=p1->next;

do

{

head=insert(head,p->expo,p->coef);

p=p->next;

}while(p!=p1->next);

p=p2->next;

do

{

head=insert(head,p->expo,p->coef);

p=p->next;

}while(p!=p2->next);

return(head);

}

int eval(node \*head)

{

node \*p;

int x,ans=0;

printf("\n Enter the value of x = ");

scanf("%d",&x);

p=head->next;

do

{

ans=ans+p->coef\*pow(x,p->expo);

p=p->next;

}while(p!=head->next);

return(ans);

}

void display(node \*head)

{

node \*p,\*q;

int n=0;

q=head->next;

p=head->next;

do

{

n++;

q=q->next;

}while(q!=head->next);

printf("\n\t The polynomial is==> ");

do

{

if(n-1)

{

printf("%dx^(%d) + ",p->coef,p->expo);

p=p->next;

}

else

{

printf(" %dx^(%d)",p->coef,p->expo);

p=p->next;

}

n--;

}while(p!=head->next);

}

void main()

{

int a,x,ch;

node \*p1,\*p2,\*p3;

p1=p2=p3=NULL;

clrscr();

while(1)

{

printf("\n\t----------------<< MENU >>---------------");

printf("\n\t Polynomial Operations :");

printf(" 1.Add");

printf("\n\t\t\t\t 2.Evaluate");

printf("\n\t\t\t\t 3.Exit");

printf("\n\t------------------------------------------- ");

printf("\n\n\n\t Enter your choice==>");

scanf("%d",&ch);

switch(ch)

{

case 1:

p1=create();

display(p1);

p2=create();

display(p2);

p3=add(p1,p2);

display(p3);

break;

case 2:

p1=create();

display(p1);

a=eval(p1);

printf("\n\n Value of polynomial=%d",a);

break;

case 3:

exit(0);

break;

default :

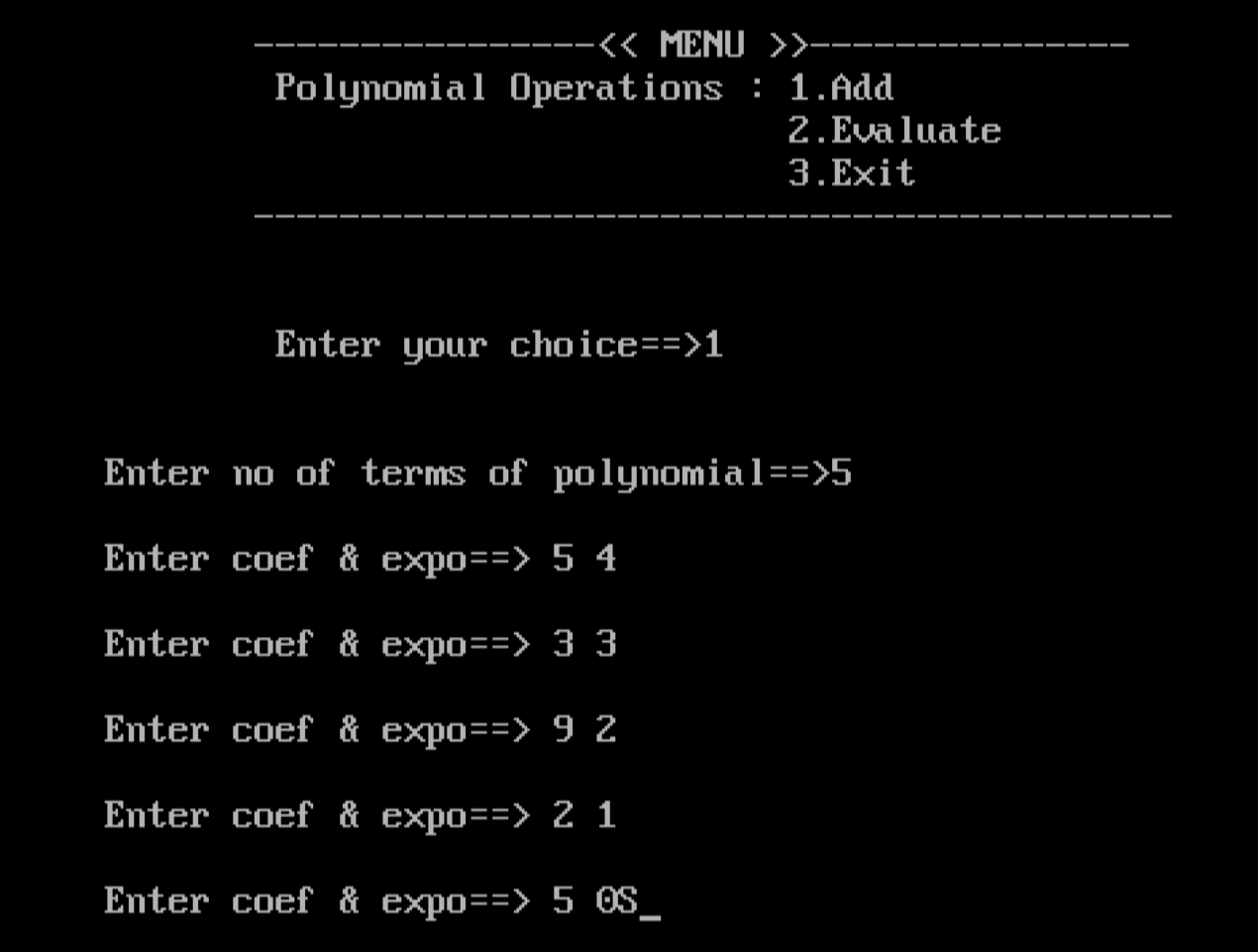
printf("\n\n\t Invalid choice");

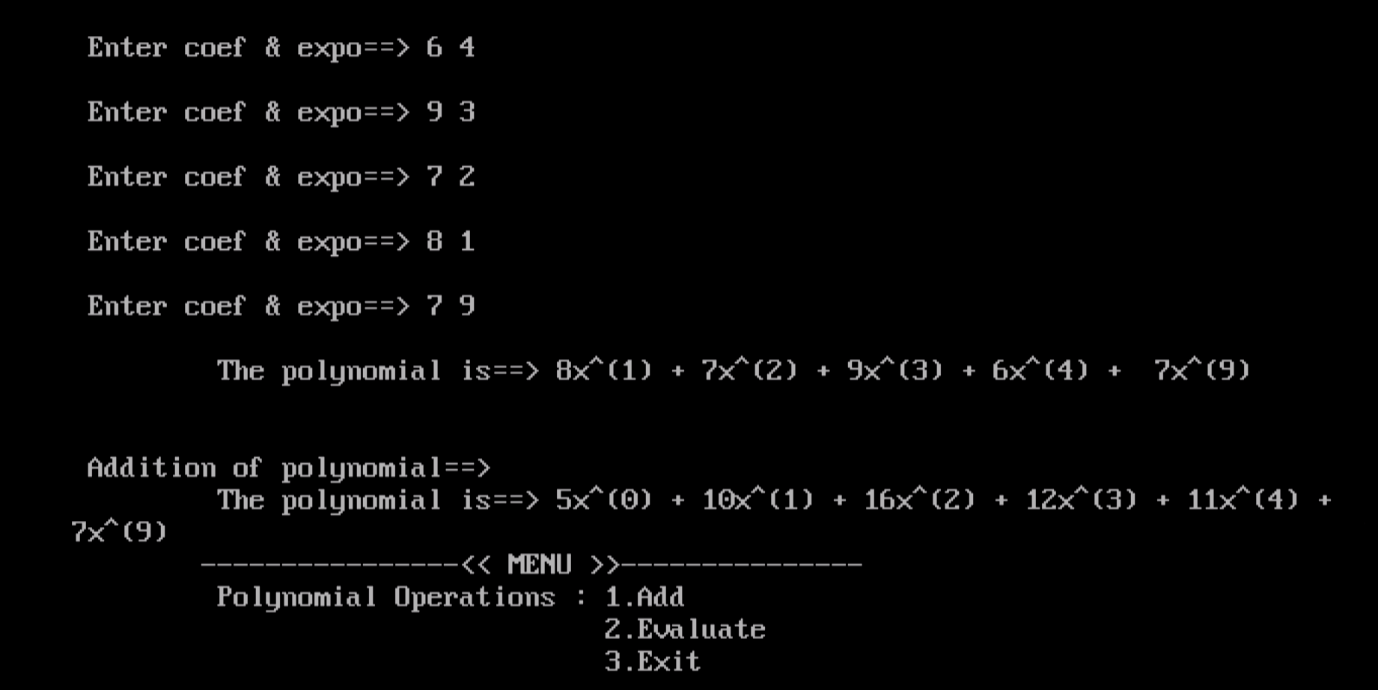
break;

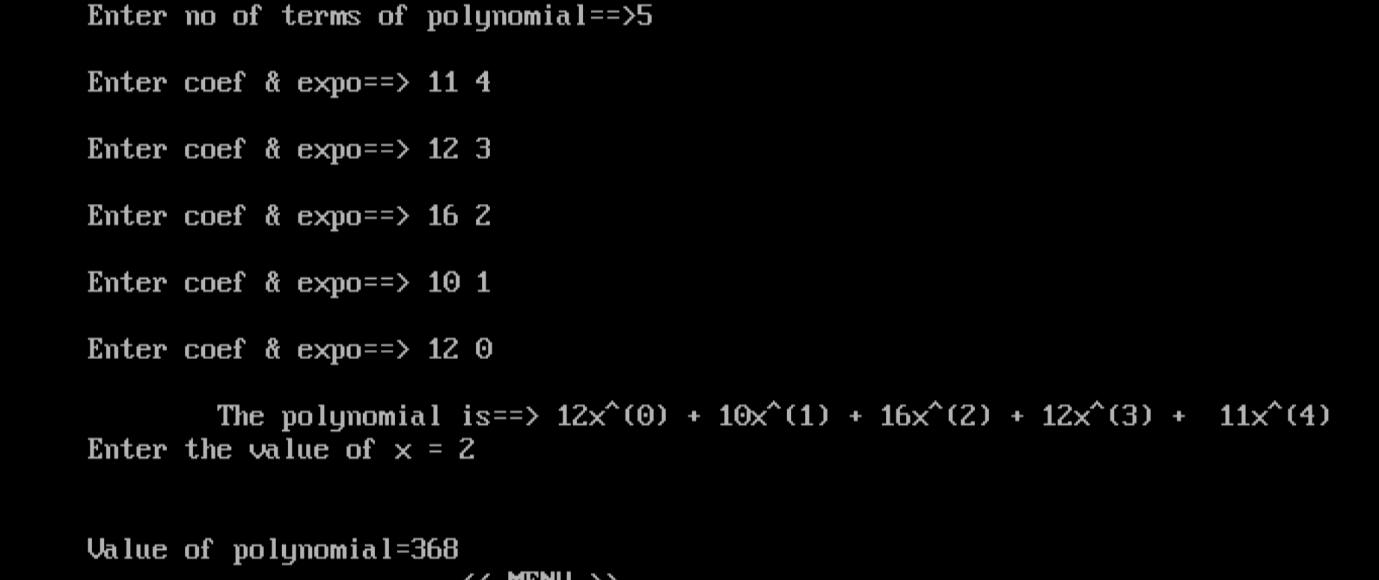
}

}

}







**Lab 10 :**

Develop a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers .

1. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2
2. Traverse the BST in Inorder, Preorder and Post Order
3. Search the BST for a given element (KEY) and report the appropriate message

Exit

#include <stdio.h>

#include <stdlib.h>

struct BST

{

int data;

struct BST \*left;

struct BST \*right;

};

typedef struct BST NODE;

NODE \*node;

NODE\* createtree(NODE \*node, int data)

{

if (node == NULL)

{

NODE \*temp;

temp= (NODE\*)malloc(sizeof(NODE));

temp->data = data;

temp->left = temp->right = NULL;

return temp;

}

if (data < (node->data))

{

node->left = createtree(node->left, data);

}

else if (data > node->data)

{

node -> right = createtree(node->right, data);

}

return node;

}

NODE\* search(NODE \*node, int data)

{

if(node == NULL)

printf("\nElement not found");

else if(data < node->data)

{

node->left=search(node->left, data);

}

else if(data > node->data)

{

node->right=search(node->right, data);

}

else

printf("\nElement found is: %d", node->data);

return node;

}

void inorder(NODE \*node)

{

if(node != NULL)

{

inorder(node->left);

printf("%d\t", node->data);

inorder(node->right);

}

}

void preorder(NODE \*node)

{

if(node != NULL)

{

printf("%d\t", node->data);

preorder(node->left);

preorder(node->right);

}

}

void postorder(NODE \*node)

{

if(node != NULL)

{

postorder(node->left);

postorder(node->right);

printf("%d\t", node->data);

}

}

NODE\* findMin(NODE \*node)

{

if(node==NULL)

{

return NULL;

}

if(node->left)

return findMin(node->left);

else

return node;

}

NODE\* del(NODE \*node, int data)

{

NODE \*temp;

if(node == NULL)

{

printf("\nElement not found");

}

else if(data < node->data)

{

node->left = del(node->left, data);

}

else if(data > node->data)

{

node->right = del(node->right, data);

}

else

{

/\* Now We can delete this node and replace with either minimum element in the right sub tree or

maximum element in the left subtree \*/

if(node->right && node->left)

{

/\* Here we will replace with minimum element in the right sub tree \*/

temp = findMin(node->right);

node -> data = temp->data;

/\* As we replaced it with some other node, we have to delete that node \*/

node -> right = del(node->right, temp->data);

}

else

{

/\* If there is only one or zero children then we can directly remove it from the tree and connect its

parent to its child \*/

temp = node;

if(node->left == NULL)

node = node->right;

else if(node->right == NULL)

node = node->left;

free(temp); /\* temp is longer required \*/

}

}

return node;

}

void main()

{

int data, ch, i, n;

NODE \*root=NULL;

clrscr();

while (1)

{

printf("\n1.Insertion in Binary Search Tree");

printf("\n2.Search Element in Binary Search Tree");

printf("\n3.Delete Element in Binary Search Tree");

printf("\n4.Inorder\n5.Preorder\n6.Postorder\n7.Exit");

printf("\nEnter your choice: "); scanf("%d", &ch);

switch (ch)

{

case 1:

printf("\nEnter N value: " );

scanf("%d", &n);

printf("\nEnter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)\n");

for(i=0; i<n; i++)

{

scanf("%d", &data);

root=createtree(root, data);

}

break;

case 2:

printf("\nEnter the element to search: ");

scanf("%d", &data);

break;

case 3:

printf("\nEnter the element to delete: ");

scanf("%d", &data);

root=del(root, data);

break;

case 4:

printf("\nInorder Traversal: \n");

inorder(root);

break;

case 5:

printf("\nPreorder Traversal: \n");

preorder(root);

break;

case 6:

printf("\nPostorder Traversal: \n");

postorder(root);

break;

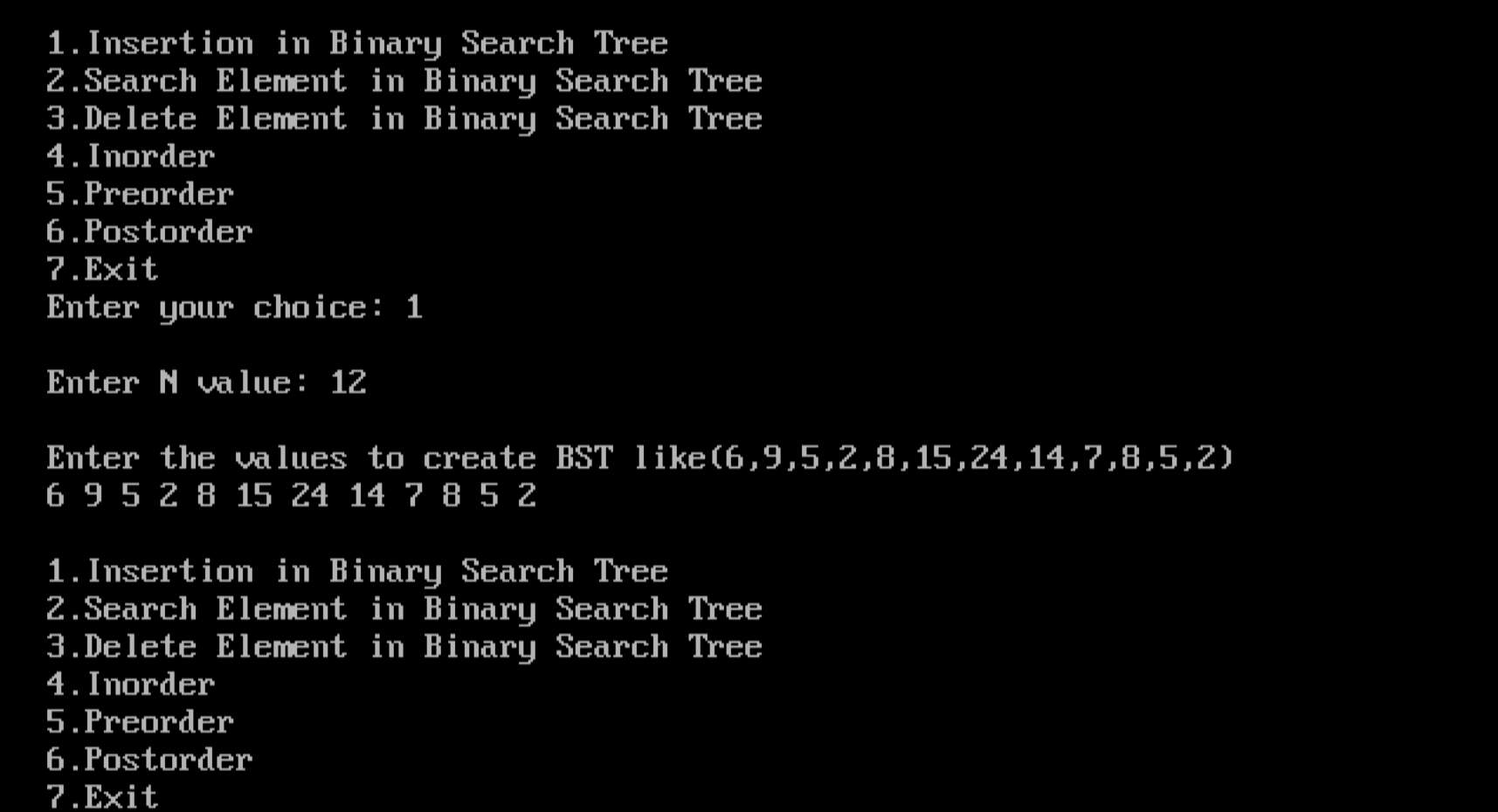
case 7:

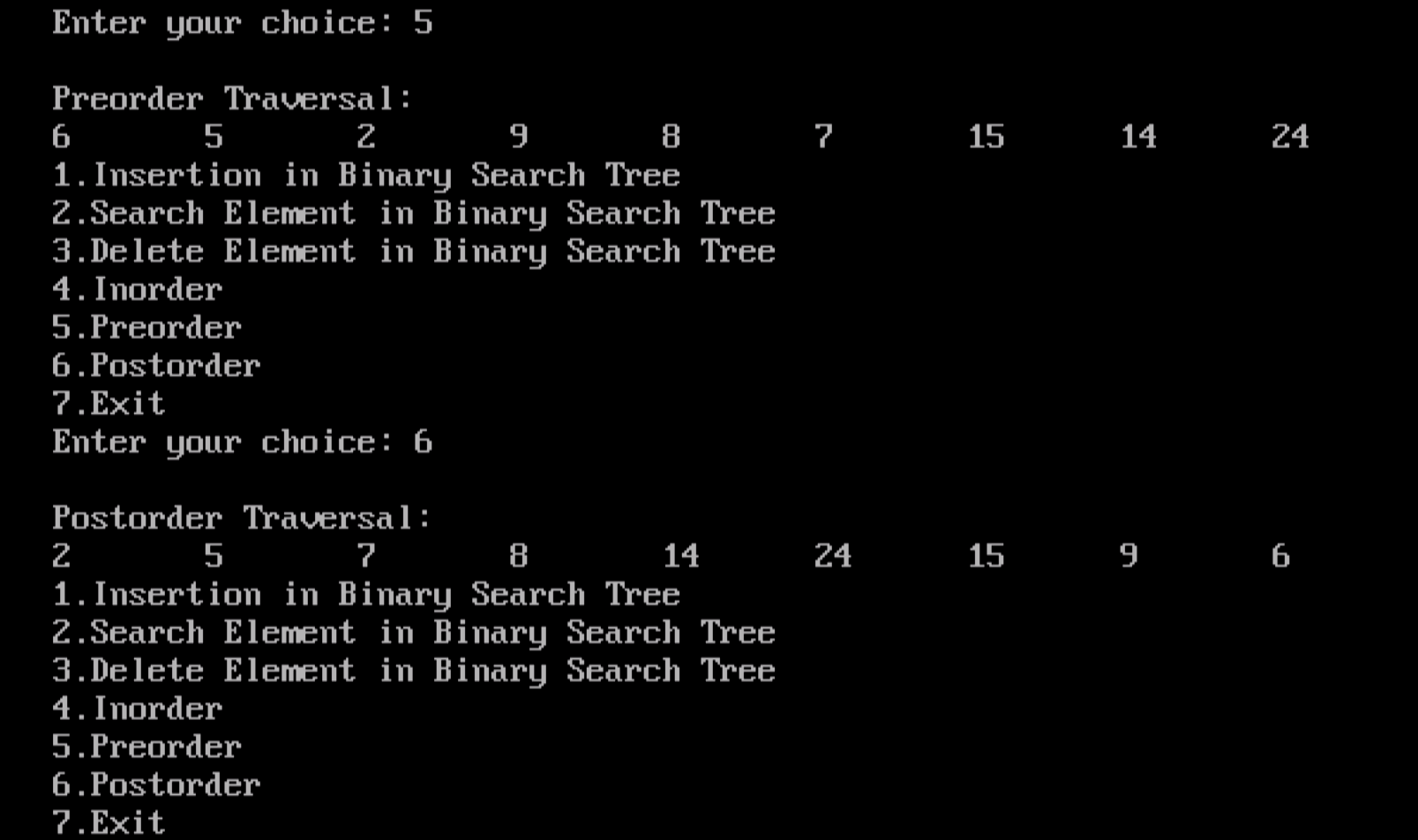
exit(0);

default : printf("\nWrong option");

break;

}}}

******



**Lab 11** :

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

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

#include <stdio.h>

#include <stdlib.h>

int a[20][20],q[20],visited[20],reach[10],n,i,j,f=0,r=-1,count=0;

void bfs(int v)

{

for(i=1;i<=n;i++)

if(a[v][i] && !visited[i])

q[++r]=i;

if(f<=r)

{

visited[q[f]]=1;

bfs(q[f++]);

}

}

void dfs(int v)

{

int i; reach[v]=1;

for(i=1;i<=n;i++)

{

if(a[v][i] && !reach[i])

{

printf("\n %d->%d",v,i);

count++;

dfs(i);

}

}

}

void main()

{

int v, choice;

printf("\n Enter the number of vertices:");

scanf("%d",&n);

for(i=1;i<=n;i++)

{

q[i]=0;

visited[i]=0;

}

for(i=1;i<=n-1;i++)

reach[i]=0;

printf("\n Enter graph data in matrix form:\n");

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

scanf("%d",&a[i][j]);

printf("1.BFS\n 2.DFS\n 3.Exit\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\n Enter the starting vertex:");

scanf("%d",&v);

bfs(v);

if((v<1)||(v>n))

{

printf("\n Bfs is not possible");

}

else

{

printf("\n The nodes which are reachable from %d:\n",v);

for(i=1;i<=n;i++)

if(visited[i])

printf("%d\t",i);

}

break;

case 2:

dfs(1);

if(count==n-1)

printf("\n Graph is connected");

else

printf("\n Graph is not connected");

break;

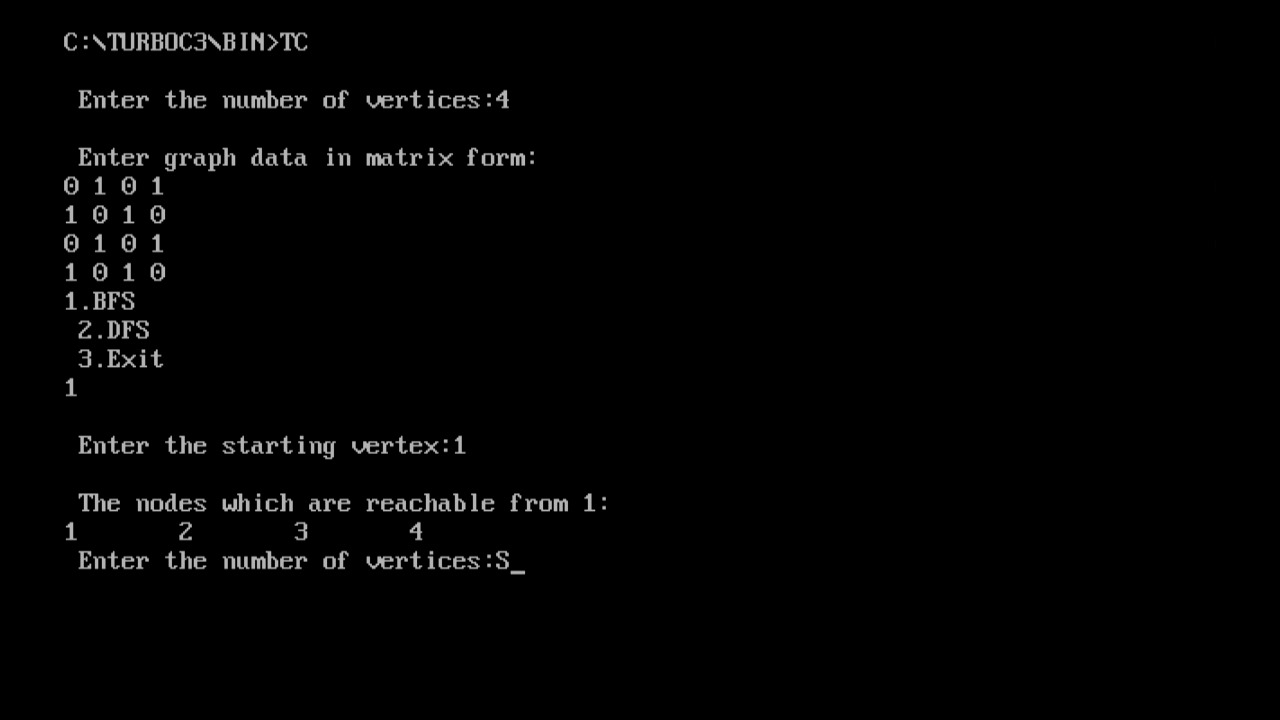
case 3:

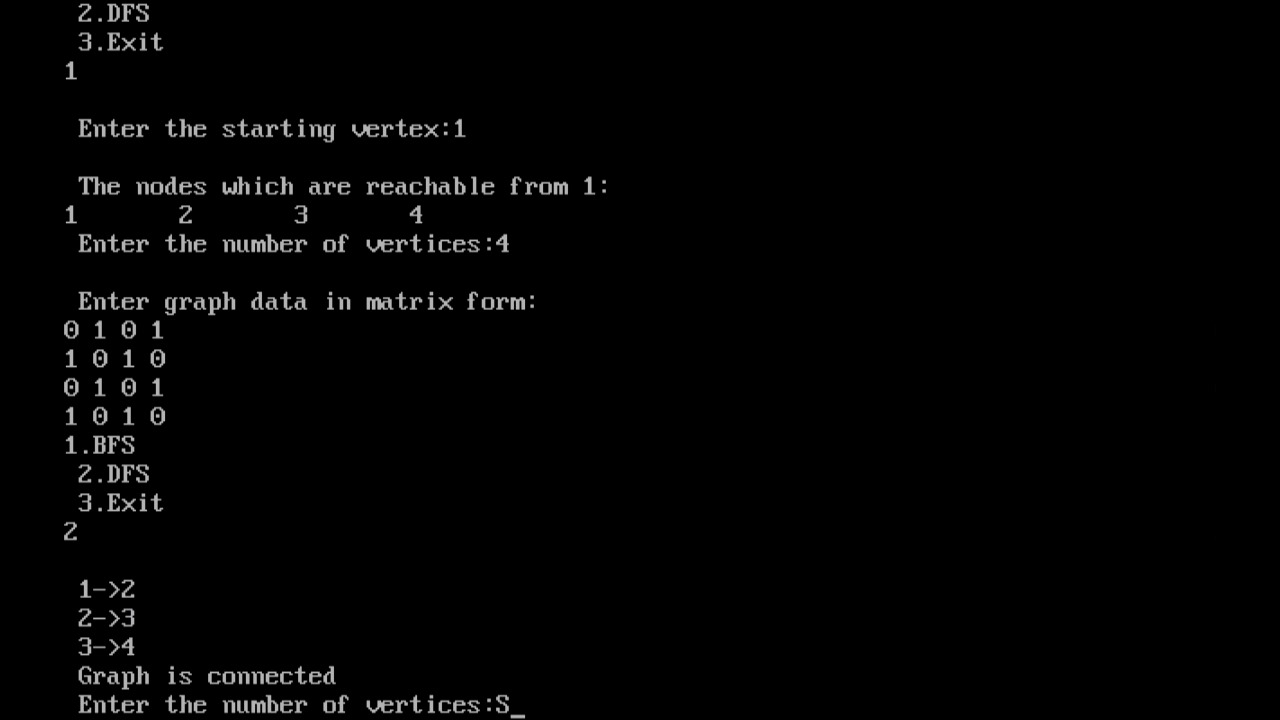
exit(0);

}

getch();

}





**Lab 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 inear probing.

#include <stdio.h>

#include <stdlib.h>

#define MAX 20

int create(int);

void display (int[]);

void linear\_prob(int[], int,int);

void main()

{

int a[MAX],num,key,i;

int ans=1;

clrscr();

printf(" collision handling by linear probing : \n");

for (i=0;i<MAX;i++)

{

a[i] = -1;

}

do

{

printf("\n Enter the data");

scanf("%d", &num);

key=create(num);

linear\_prob(a,key,num);

printf("\n Do you wish to continue ? (1/0) ");

scanf("%d",&ans);

}while(ans);

display(a);

getch();

}

int create(int num)

{

int key;

key=num%20;

return key;

}

void linear\_prob(int a[MAX], int key, int num)

{

int flag, i, count=0;

flag=0;

if(a[key]== -1)

{

a[key] = num;

}

else

{

printf("\nCollision Detected...!!!\n");

i=0;

while(i<MAX)

{

if (a[i]!=-1)

count++;

i++;

}

printf("Collision avoided successfully using LINEAR PROBING\n");

if(count == MAX)

{

printf("\n Hash table is full");

display(a);

exit(1);

}

for(i=key+1; i<MAX; i++)

if(a[i] == -1)

{

a[i] = num;

flag =1;

break;

}

i=0;

while((i<key) && (flag==0))

{

if(a[i] == -1)

{

a[i] = num;

flag=1;

break;

}

i++;

}

}

}

void display(int a[MAX])

{

int i,choice;

printf("1.Display ALL \n2.Filtered Display \n");

scanf("%d",&choice);

if(choice==1)

{

printf("\n The hash table is\n");

for(i=0; i<MAX; i++)

printf("\n %d %d ", i, a[i]);

}

else

{

printf("\n The hash table is\n");

for(i=0; i<MAX; i++)

if(a[i]!=-1)

{

printf("\n %d %d ", i, a[i]);

continue;

}

}

}

