PROGRAM 1

```
#include <stdio.h>
#include<stdlib.h>
struct Day {
char * dayName;
int date;
char * activity;};
void create(struct Day * day) {
day -> dayName = (char * )
malloc(sizeof(char) * 20);
day -> activity = (char * ) malloc(sizeof(char) * 100);
printf("Enter the day name:");
scanf("%s", day -> dayName);
printf("Enter the date:");
scanf("%d", & day -> date);
printf("Enter the activity for the day:");
scanf(" %[^\n]s", day -> activity); }
void read(struct Day * calendar, int size) {
for (int i = 0; i < size; i++) {
printf("Enter details for Day %d:\n", i + 1);
create( & calendar[i]); } }
void display(struct Day * calendar, int size) {
printf("\nWeek's Activity Details:\n");
for (int i = 0; i < size; i++) {
printf("Day %d:\n", i + 1);
printf("Day Name: %s\n", calendar[i].dayName);
printf("Date: %d\n", calendar[i].date);
printf("Activity: %s\n", calendar[i].activity);} }
void freeMemory(struct Day * calendar, int size) {
for (int i = 0; i < size; i++) {
free(calendar[i].dayName);
free(calendar[i].activity);
} } int main() {
int size;
printf("Enter the number of days in the week:");
scanf("%d", & size);
struct Day * calendar = (struct Day * )
malloc(sizeof(struct Day) * size);
if (calendar == NULL) {
printf("Memory allocation failed. Exiting program.\n");
return 1; }
read(calendar, size);
display(calendar, size);
freeMemory(calendar, size);
free(calendar);
return 0; }
```

PROGRAM 2

```
#include<stdio.h>
char str[50], pat[20], rep[20], res[50];
int c = 0, m = 0, i = 0, j = 0, k, flag = 0;
void stringmatch() {
while (str[c] != '\0') {
if (str[m] == pat[i]) {
i++; m++;
if (pat[i] == '\0') {
flag = 1;
for (k = 0; rep[k] != '\0'; k++, j++) {
res[j] = rep[k]; }
i = 0; c = m; } }
else {res[j] = str[c];
j++; c++; m = c; i = 0; } }
res[j] = '\0'; }
void main() {
printf("Enter the main string:");
gets(str);
printf("\nEnter the pat string:");
gets(pat);
printf("\nEnter the replace string:");
gets(rep);
printf("\nThe string before pattern match is:\n %s", str);
stringmatch();
if (flag == 1)
printf("\nThe string after pattern match and replace is: \n %s ", res);
else
printf("\nPattern string is not found");}
```

PROGRAM 3

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 3
int s[MAX]; int top = -1;
void push(int item);
int pop();
void palindrome();
void display();
void main() {
int choice, item;
while (1) {
printf("\n\n\n\n~~~~~~Menu~~~~~~ : ");
printf("\n=>1.Push an Element to Stack and Overflow demo ");
printf("\n=>2.Pop an Element from Stack and Underflow demo");
printf("\n=>3.Palindrome demo ");
printf("\n=>4.Display ");
printf("\n=>5.Exit");
printf("\nEnter your choice: ");
scanf("%d", & choice);
switch (choice) {
case 1: printf("\nEnter an element to be pushed: ");
scanf("%d", & item);
push(item); break;
case 2: item = pop();
if (item != -1)
printf("\nElement popped is: %d", item);
break;
case 3: palindrome();
break; case 4: display();
break;
case 5: exit(1);
default: printf("\nPlease enter valid choice "); break;
} } } void push(int item) {
if (top == MAX - 1) {
printf("\n~~~~Stack overflow~~~~");
return; }
top = top + 1;
s[top] = item; }
int pop() {
int item;
if (top == -1) {
```

```
printf("\n~~~~Stack underflow~~~~");
return -1; }
item = s[top];
top = top - 1;
return item; }
void display() {
int i;
if (top == -1) {
printf("\n~~~~Stack is empty~~~~");
return; }
printf("\nStack elements are:\n ");
for (i = top; i >= 0; i--)
printf("| %d |\n", s[i]); }
void palindrome() {
int flag = 1, i;
printf("\nStack content are:\n");
for (i = top; i >= 0; i--)
printf("| %d |\n", s[i]);
printf("\nReverse of stack content are:\n");
for (i = 0; i <= top; i++)
printf("| %d |\n", s[i]);
for (i = 0; i <= top / 2; i++) {
if (s[i] != s[top - i]) {
flag = 0; break; } }
if (flag == 1) {
printf("\nIt is palindrome number"); }
Else {
printf("\nIt is not a palindrome number"); } }
```

PROGRAM 4

```
#include<stdio.h>
void infix_to_postfix();
void push(char);
char pop();
int priority(char);
char infix[30], postfix[30],stack[30];
int top=-1;
void main() {
printf("Enter the valid Infix expression \n");
scanf("%s",infix);
infix_to_postfix();
printf("\n Infix expression : %s",infix);
printf("\n Postfix expression : %s\n",postfix); }
void push(char item) {
stack[++top]=item; }
char pop() {
return stack[top--]; }
int priority(char symb) {
int p;
switch(symb) {
case '+': case '-': p=1; break;
case '*': case '/': case '%': p=2; break;
case '^': case '$': p=3; break;
case '(': case ')': p=0; break;
case '#': p=-1; break; }
return p; }
void infix_to_postfix() {
int i=0,j=0;
char symb,temp;
push('#');
for(i=0;infix[i]!='\0';i++) {
symb=infix[i];
switch(symb) {
case '(': push(symb); break;
case ')': temp=pop();
while(temp!='(')
{ postfix[j++]=temp; temp=pop(); } break;
case'+': case'-': case'*': case'/': case'%': case'^': case'$':
while(priority(stack[top])>=priority(symb))
{ temp=pop();
postfix[j++]=temp; }
push(symb); break;
default: postfix[j++]=symb; }}
while(top>0)
 { temp=pop();
   postfix[j++]=temp; }
  postfix[j]='\0';}
```

PROGRAM 5

```
#include<stdio.h>
#include<math.h>
void push(float);
float pop();
void evaluate(char[]);
float stack[20];
int top=-1;
void main() {
int choice,n;
char postfix[100];
while(1)
{ printf("\n STACK APPLICATIONS");
printf("\n Enter your Choice: ");
printf("\n 1. Evaluation of postfix expression with single digit operands and operators");
printf("\n 2. Solving Tower of Hanoi problem with N disks");
printf("\n 3. Exit \n");
scanf("%d", &choice);
switch(choice) {
case 1 : printf("Enter a valid postfix expression\n");
scanf("%s",postfix);
evaluate(postfix); break;
case 2 : printf("\n Enter the number of discs:\n");
scanf("%d",&n); tower(n,'A','C','B');
printf("\n Total number of moves are %d",(int)pow(2,n)-1); break;
case 3 : return;
default : printf("\n Invalid Choice"); }}}
void push(float item) {
stack[++top]=item; }
float pop() {
return stack[top--]; }
void evaluate(char postfix[100]) {
int i; float op1, op2, res;
char symb;
for(i=0;postfix[i]!='\0';i++){
symb=postfix[i];
if(isdigit(symb))
push(symb-'0');
switch(symb) {
case '+':op2=pop(); op1=pop(); res=op1+op2; push(res); break;
case '-':op2=pop(); op1=pop(); res=op1-op2; push(res); break;
case '*':op2=pop(); op1=pop(); res=op1*op2; push(res); break;
case '/':op2=pop(); op1=pop();
if(op2==0) {
printf("Division by zero Error\n"); return; }
res=op1/op2; push(res); break;
case '%': op2=pop();
op1=pop();
if(op2==0) {
printf("Division by zero Error\n");
return; } res=(int)op1%(int)op2; push(res); break;
case '^':op2=pop(); op1=pop();
res=pow(op1,op2); push(res); break; }}
res=pop();
if(top==-1)
printf("\n Result: %f\n ",res);
else {printf("\nINVALID POSTFIX EXPRESSION\n");
top=-1; } }
void tower(int n,int source,int destination,int aux) {
if(n==0) return;
tower(n-1,source,aux,destination);
printf("\n Move disc %d from %c to %c",n,source,destination);
tower(n-1,aux,destination,source); }
```

PROGRAM 6

```
#include<stdio.h>
#include<stdlib.h>
#define max 5
int q[max],f=-1,r=-1;
void ins() {
if(f==(r+1)%max)
printf("\nQueue overflow");
else { if(f==-1) f++;
r=(r+1)%max;
printf("\nEnter element to be inserted:");
scanf("%d",&q[r]); } }
void del() {
if(r==-1)
printf("\nQueue underflow");
else {
printf("\nElemnt deleted is:%d",q[f]);
if(f==r) f=r=-1;
else f=(f+1)%max; } }
void disp() {
if(f==-1)
printf("\nQueue empty");
else {
int i;
printf("\nQueue elements are:\n");
for(i=f;i!=r;i=(i+1)%max)
printf("%d\t",q[i]);
printf("%d",q[i]);
printf("\nFront is at:%d\nRear is at:%d",q[f],q[r]); } }
int main() {
printf("\nCircular Queue operations");
printf("\n1.Insert");
printf("\n2.Delete");
printf("\n3.Display");
printf("\n4.Exit");
int ch;  do{
printf("\nEnter choice:");
scanf("%d",&ch);
switch(ch) {
case 1:ins();break;
case 2:del();break;
case 3:disp();break;
case 4:exit(0);
default:printf("\nInvalid choice...!"); } }while(1); return 0; }
```

PROGRAM 7

```
#include<stdio.h>
#include<stdlib.h>
struct node
char usn[25], name[25], branch[25];
int sem;
long int phone;
struct node * link; };
typedef struct node * NODE;
NODE start = NULL;
int count = 0;
NODE create() {
NODE snode;
snode = (NODE) malloc(sizeof(struct node));
if (snode == NULL) {
printf("\nMemory is not available");
exit(1); }
printf("\nEnter the usn,Name,Branch, sem,PhoneNo of the student:");
scanf("%s %s %s %d %ld", snode -> usn, snode -> name, snode -> branch, & snode -> sem, & snode ->
phone);
snode -> link = NULL;
count++;
return snode; }
NODE insertfront() {
NODE temp;
temp = create();
if (start == NULL) {
return temp; }
temp -> link = start;
return temp; }
NODE deletefront() {
NODE temp;
if (start == NULL) {
printf("\nLinked list is empty");
return NULL; }
if (start -> link == NULL) {
printf("\nThe Student node with usn:%s is deleted ", start -> usn);
count--;
free(start);
return NULL; }
temp = start;
start = start -> link;
```

```
printf("\nThe Student node with usn:%s is deleted", temp -> usn);
count--; free(temp);
return start; }
NODE insertend() {
NODE cur, temp;
temp = create();
if (start == NULL) {
return temp; }
cur = start;
while (cur -> link != NULL) {
cur = cur -> link; }
cur -> link = temp;
return start; }
NODE deleteend() {
NODE cur, prev;
if (start == NULL) {
printf("\nLinked List is empty"); return NULL; }
if (start -> link == NULL) {
printf("\nThe student node with the usn:%s is deleted",start -> usn);
free(start);
count--; return NULL; }
prev = NULL;
cur = start;
while (cur -> link != NULL) {
prev = cur; cur = cur -> link; }
printf("\nThe student node with the usn:%s is deleted", cur -> usn);
free(cur); prev -> link = NULL;
count--;
return start; }
void display() {
NODE cur;
int num = 1;
if (start == NULL) {
printf("\nNo Contents to display in SLL \n");
return; }
printf("\nThe contents of SLL: \n");
cur = start;
while (cur != NULL) {
printf("\n||%d|| USN:%s| Name:%s| Branch:%s| Sem:%d| Ph:%ld|", num, cur -> usn, cur -> name, cur ->
branch, cur -> sem, cur -> phone); cur = cur -> link; num++; }
printf("\n No of student nodes is %d \n", count); }
void stackdemo() {
```

```
int ch; while (1) {
printf("\n~~~Stack Demo using SLL~~~\n");
printf("\n1:Push operation \n2: Pop operation \n3: Display \n4:Exit \n");
printf("\nEnter your choice for stack demo:");
scanf("%d", & ch);
switch (ch) {
case 1: start = insertfront();break;
case 2: start = deletefront(); break;
case 3: display(); break;
default: return; } } return; }
int main() {
int ch, i, n;
while (1) {
printf("\n~~~Menu~~~");
printf("\nEnter your choice for SLL operation \n");
printf("\n1:Create SLL of Student Nodes");
printf("\n2:DisplayStatus");
printf("\n3:InsertAtEnd");
printf("\n4:DeleteAtEnd");
printf("\n5:Stack Demo using SLL(Insertion and Deletion at Front)");
printf("\n6:Exit \n");
printf("\nEnter your choice:");
scanf("%d", & ch);
switch (ch) {
case 1: printf("\nEnter the no of students: ");
scanf("%d", & n);
for (i = 1; i <= n; i++)
start = insertfront();break;
case 2: display(); break;
case 3: start = insertend(); break;
case 4: start = deleteend(); break;
case 5: stackdemo(); break;
case 6: exit(0); default: printf("\nPlease enter the valid choice");} } }
```

PROGRAM 8

```
#include<stdio.h>
#include<stdlib.h>
struct node {
char ssn[25], name[25], dept[10], designation[25];
int sal;
long int phone;
struct node * llink;
struct node * rlink; };
typedef struct node * NODE;
NODE first = NULL;
int count = 0;
NODE create() {
NODE enode;
enode = (NODE) malloc(sizeof(struct node));
if (enode == NULL) {
printf("\nRunning out of memory");
exit(0); }
printf("\nEnter the ssn, Name, Department, Designation, Salary, PhoneNo of the employee:\n"); scanf("%s
%s %s %s %d %ld", enode -> ssn, enode -> name, enode -> dept, enode -> designation, & enode -> sal, &
enode -> phone);
enode -> llink = NULL;
enode -> rlink = NULL;
count++; return enode; }
NODE insertfront() {
NODE temp;
temp = create();
if (first == NULL) {
return temp; }
temp -> rlink = first;
first -> llink = temp;
return temp; }
void display() {
NODE cur;
int nodeno = 1;
cur = first;
if (cur == NULL)
printf("\nNo Contents to display in DLL");
while (cur != NULL) {
printf("\nENode:%d||SSN:%s|Name:%s|Department:%s| Designation:%s|Salary:%d|Phone no:%ld",
nodeno, cur -> ssn, cur -> name, cur -> dept, cur -> designation, cur -> sal, cur -> phone); cur = cur -> rlink;
nodeno++; }
```

```
printf("\nNo of employee nodes is %d", count); }
NODE deletefront() {
NODE temp;
if (first == NULL) {
printf("\nDoubly Linked List is empty");
return NULL; }
if (first -> rlink == NULL) {
printf("\nThe employee node with the ssn:%s is deleted", first -> ssn);
free(first);
count--;
return NULL; }
temp = first;
first = first -> rlink;
temp -> rlink = NULL; first -> llink = NULL;
printf("\nThe employee node with the ssn:%s is deleted", temp -> ssn);
free(temp); count--;
return first; }
NODE insertend() {
NODE cur, temp;
temp = create();
if (first == NULL) {
return temp; }
cur = first;
while (cur -> rlink != NULL) {
cur = cur -> rlink; }
cur -> rlink = temp;
temp -> llink = cur;
return first; }
NODE deleteend() {
NODE prev, cur;
if (first == NULL) {
printf("\nDoubly Linked List is empty"); return NULL; }
if (first -> rlink == NULL) {
printf("\nThe employee node with the ssn:%s is deleted", first -> ssn);
free(first);
count--; return NULL; }
prev = NULL;
cur = first;
while (cur -> rlink != NULL) {
prev = cur; cur = cur -> rlink; }
cur -> llink = NULL;
printf("\nThe employee node with the ssn:%s is deleted", cur -> ssn);
```

```
free(cur); prev -> rlink = NULL;
count--; return first; }
void deqdemo() {
int ch; while (1) {
printf("\nDemo Double Ended Queue Operation");
printf("\n1:InsertQueueFront\n 2: DeleteQueueFront\n 3:InsertQueueRear\n 4:DeleteQueueRear\n
5:DisplayStatus\n 6: Exit \n");
scanf("%d", & ch);
switch (ch) {
case 1: first = insertfront(); break;
case 2: first = deletefront(); break;
case 3: first = insertend(); break;
case 4: first = deleteend(); break;
case 5: display(); break;
default: return; } } }
void main() {
int ch, i, n;
while (1) {
printf("\n\n~~~Menu~~~");
printf("\n1:Create DLL of Employee Nodes");
printf("\n2:DisplayStatus");
printf("\n3:InsertAtEnd");
printf("\n4:DeleteAtEnd");
printf("\n5:InsertAtFront");
printf("\n6:DeleteAtFront");
printf("\n7:Double Ended Queue Demo using DLL");
printf("\n8:Exit \n");
printf("\nPlease enter your choice: ");
scanf("%d", & ch); switch (ch) {
case 1: printf("\nEnter the no of Employees: ");
scanf("%d", & n);
for (i = 1; i <= n; i++)
first = insertend(); break;
case 2: display(); break;
case 3: first = insertend(); break;
case 4: first = deleteend(); break;
case 5: first = insertfront(); break;
case 6: first = deletefront(); break;
case 7: deqdemo(); break;
case 8: exit(0);
default: printf("\nPlease Enter the valid choice"); } } }
```

PROGRAM 9

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
typedef struct node1 {
int coef, po_x,po_y,po_z;
struct node1 *next; };
typedef struct node1* node;
node insert(node head,int coef,int po_x, int po_y, int po_z) {
node p,q;
p=(node)malloc(size of(struct node1));
p->coef=coef;
p- >po_x=po_ x;
p- >po_y=po_ y;
p- >po_z=po_ z;
p- >next=NUL L;
q= head;
while(q- >next!=head )
q=q->next;
q->next=p; p->next=head;
return head; }
void display(node head) {
if(head->next==head) {
printf("Polynomial is enmpty");
return; }
node cur;
cur=head- >next;
do { printf("%dx^ %dy^%dz^%d",cur->coef,cur- >po_x,cur->po_y,cur->po_z);
cur=cur- >next; if(cur!=head) {
printf(" + "); } }while(cur!=h ead);
printf("\n"); }
int eval(node head, int x, int y , int z) {
int ans=0, term;
if(head->next==head) {
return ans; }
node cur= head->next;
do {
term=cur->coef;
term *= pow(x,cur- >po_x); term *= pow(y,cur- >po_y);
term *= pow(z,cur- >po_z); ans+=term; cur=cur->next; }
while(cur!=head); return ans; }
bool match(node p1, node p2) {
```

```
bool flag=true;
if(p1->po_x!=p2->po_x)
flag=false;
if(p1->po_y!=p2->po_y)
flag=false;
if(p1->po_z!=p2->po_z)
flag=false;
return flag; }
node add(node p1,node p2, node res) {
node cur1, cur2; cur1=p1- >next;
cur2=p2- >next; do {
res=insert(res, cur1->coef,cur1- >po_x,cur1->po_y,cur1->po_z);
cur1=cur1->next; }
while(cur1!=p1); do {
cur1=res- >next;
bool flag1=false; do {
if(match(cur1,cur2)) {
cur1- >coef+=cur2- >coef;
flag1=true; break; }
cur1=cur1->next; }
while(cur1!=res);
if(!flag1)
res=insert(res,cur2->coef,cur2- >po_x,cur2->po_y,cur2->po_z);
cur2=cur2- >next; }while(cur2!=p2);
return res; }
int main() {
node p1=(node)malloc(sizeof( struct node1));
p1- >next=p1; node p2= (node)malloc(sizeof(stru ct node1));
p2->next=p2; node res=(node)malloc(sizeof(struct node1));
res->next=res; p1=insert(p1,+6,2,2,1); p1=insert(p1,-4,0,1,5);
p1=insert(p1,+3,3,1,1); p1=insert(p1,+2,1,5,1);
p1=insert(p1,-2,1,1,3); printf("p1( X,Y,Z)= \n"); display(p1) ;
p2=insert(p2,+1,1,1,1); p2=insert(p2,+4,3,1,1);
printf("p2 (x,y,z)= \n"); display(p2) ;
res=add(p1,p2,res);
printf("Addition of two polynomials : res(x,y,z)= \n");
display(res); int x,y,z; x=1; y=2; z=3; int ires;
ires=eval (res, x, y ,z);
printf("\n res of polysum(%d,%d,%d): %d) \n ", x,y,z, ires);
return 0; }
```

PROGRAM 10

```
#include<stdio.h>
#include<stdlib.h>
struct BST
{ int data;
struct BST * lchild;
struct BST * rchild; };
typedef struct BST * NODE; NODE create() {
NODE temp; temp = (NODE) malloc(sizeof(struct BST));
printf("\nEnter The value: ");
scanf("%d", & temp -> data);
temp -> lchild = NULL; temp -> rchild = NULL; return temp; }
void insert(NODE root, NODE newnode);
void inorder(NODE root);
void preorder(NODE root);
void postorder(NODE root);
void search(NODE root);
void insert(NODE root, NODE newnode) {
if (newnode -> data < root -> data) {
if (root -> lchild == NULL) root -> lchild = newnode;
else insert(root -> lchild, newnode); }
if (newnode -> data > root -> data) {
if (root -> rchild == NULL) root -> rchild = newnode;
else insert(root -> rchild, newnode); } }
void search(NODE root) {
int key; NODE cur; if (root == NULL)
{ printf("\nBST is empty."); return; }
printf("\nEnter Element to be searched: ");
scanf("%d", & key); cur = root;
while (cur != NULL) { if (cur -> data == key) {
printf("\nKey element is present in BST"); return; }
if (key < cur -> data) cur = cur -> lchild;
else cur = cur -> rchild; }
printf("\nKey element is not found in the BST"); }
void inorder(NODE root) {
if (root != NULL) { inorder(root -> lchild);
printf("%d ", root -> data); inorder(root -> rchild); } }
void preorder(NODE root) { if (root != NULL) {
printf("%d ", root -> data);
preorder(root -> lchild); preorder(root -> rchild); } }
void postorder(NODE root) { if (root != NULL) {
postorder(root -> lchild);
postorder(root -> rchild);
printf("%d ", root -> data); } }
void main() {
int ch, key, val, i, n;
NODE root = NULL, newnode;
while (1) {
printf("\n~~~~BST MENU~~~~");
printf("\n1.Create a BST");
printf("\n2.Search");
printf("\n3.BST Traversals: ");
printf("\n4.Exit");
printf("\nEnter your choice: ");
scanf("%d", & ch);
switch (ch) {
case 1: printf("\nEnter the number of elements: ");
scanf("%d", & n);
for (i = 1; i <= n; i++) {
newnode = create();
if (root == NULL) root = newnode;
else insert(root, newnode); } break;
case 2: if (root == NULL) printf("\nTree Is Not Created");
else {
printf("\nThe Preorder display : ");
preorder(root);
printf("\nThe Inorder display : ");
inorder(root); printf("\nThe Postorder display : ");
postorder(root); } break;
case 3: search(root); break;
case 4: exit(0);}}}
```

PROGRAM 11

```
#include<stdio.h>
#include<stdlib.h>
int a[50][50], n, visited[50];
int q[20], front = -1, rear = -1;
int s[20], top = -1, count = 0;
void bfs(int v) {
int i, cur; visited[v] = 1;
q[++rear] = v;
while (front != rear) {
cur = q[++front];
for (i = 1; i <= n; i++) {
if ((a[cur][i] == 1) && (visited[i] == 0)) {
q[++rear] = i;
visited[i] = 1;
printf("%d ", i); } } } }
void dfs(int v) {
int i; visited[v] = 1;
s[++top] = v;
for (i = 1; i <= n; i++) {
if (a[v][i] == 1 && visited[i] == 0) {
printf("%d ", i); dfs(i); } } }
int main() {
int ch, start, i, j;
printf("\nEnter the number of vertices in graph:");
scanf("%d", & n);
printf("\nEnter the adjacency matrix:\n");
for (i = 1; i <= n; i++) {
for (j = 1; j <= n; j++)
scanf("%d", & a[i][j]); }
for (i = 1; i <= n; i++) visited[i] = 0;
printf("\nEnter the starting vertex: ");
scanf("%d", & start);
printf("\n==>1. BFS: Print all nodes reachable from a given starting node");
printf("\n==>2. DFS: Print all nodes reachable from a given starting node");
printf("\n==>3:Exit");
printf("\nEnter your choice: ");
scanf("%d", & ch);
switch (ch) {
case 1: printf("\nNodes reachable from starting vertex %d are: ", start);
bfs(start);
for (i = 1; i <= n; i++) {
```

```
if (visited[i] == 0)
printf("\nThe vertex that is not reachable is %d", i); } break;
case 2: printf("\nNodes reachable from starting vertex %d are:\n", start);
dfs(start); break;
case 3: exit(0);
default: printf("\nPlease enter valid choice:"); } }
```

PROGRAM 12

```
#include<stdio.h>
#include<stdlib.h>
int key[20], n, m; int * ht, index;
int count = 0;
void insert(int key) {
index = key % m;
while (ht[index] != -1) {
index = (index + 1) % m; }
ht[index] = key; count++; }
void display() {
int i; if (count == 0) {
printf("\nHash Table is empty"); return; }
printf("\nHash Table contents are:\n ");
for (i = 0; i < m; i++)
printf("\n T[%d] --> %d ", i, ht[i]); }
void main() {
int i;
printf("\nEnter the number of employee records (N) :");
scanf("%d", & n);
printf("\nEnter the two digit memory locations (m) for hash table:");
scanf("%d", & m);
ht = (int * ) malloc(m * sizeof(int));
for (i = 0; i < m; i++)
ht[i] = -1;
printf("\nEnter the four digit key values (K) for N Employee Records:\n ");
for (i = 0; i < n; i++)
scanf("%d", & key[i]);
for (i = 0; i < n; i++) {
if (count == m) {
printf("\n~~~Hash table is full. Cannot insert the record %d key~~~", i + 1);
break; }
insert(key[i]); }
display(); }
```