**PROGRAM 1**

**Design, Develop and Implement a menu driven Program in C for the following Array operations**

1. **Creating an Array of N Integer Elements**
2. **Display of Array Elements with Suitable Headings**
3. **Inserting an Element (ELEM) at a given valid Position (POS)**
4. **Deleting an Element at a given valid Position (POS)**
5. **Exit.**

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

**Source Code:**

/\*

1. Design, Develop and Implement a menu driven Program in C for the following Array operations

a. Creating an Array of N Integer Elements

b. Display of Array Elements with Suitable Headings

c. Inserting an Element (ELEM) at a given valid Position (POS)

d. Deleting an Element at a given valid Position(POS)

e. Exit.

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

\*/

#include<stdio.h>

#include<conio.h>

#define SIZE 15

void create(int array[], int n)

{

int i;

printf(" Enter n elemnts\n");

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

{

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

}

return ;

}

void display(int array[], int n)

{ int i;

printf(" Elemnets of the array are\n");

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

{

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

}

printf("\n");

}

int insertAtPos(int array[], int n, int position, int value)

{ int i;

for (i = n ; i >= position; i--)

array[i+1] = array[i];

array[position] = value;

n++; return n;

}

int deleteFromPos(int array[], int n, int position)

{ int i, v;

v=array[position];

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

array[i] = array[i+1];

n--; return n;

}

int main()

{

int array[SIZE], n, choice, flag=0;

int position, value; int count=0;

char answer;

while(1)

{

printf("1. Create\n");

printf("2. Display\n");

printf("3. Inserting Element at given valid position\n");

printf("4.Delete an Element at a given valid Position \n”);

printf("5. Exit\n");

printf("Enter choice =");

scanf("%d", &choice);

switch(choice)

{

case 1 : if(flag==0)

{

flag=1;

printf("Enter no. of elements=");

scanf("%d", &n);

// if n is > SIZE. Reduce n=SIZE

if(n>SIZE) n=SIZE;

create(array,n);

}

else

{

printf("Array is already created.....");

}

break;

case 2 : display(array, n);

break;

case 3 : printf("Enter the location to insert element=\n");

scanf("%d", &position);

if(position>=SIZE || position>n+1)

{ printf("IT is not valid Position");

break;

}

printf("Enter the value to insert=\n");

scanf("%d", &value);

n=insertAtPos(array, n, position, value);

break;

case 4 :printf("Enter the location to delete element\n");

scanf("%d", &position);

if(position>n)

{

printf("Postion is beyond the array element\n");

break;

}

if(n==0)

{

printf("Array is empty\n"); break;

}

n=deleteFromPos(array, n, position);

break;

case 5 : return 0;

default : printf(" Please enter correct choice");

break;

}

getch();

}

return 0;

}

**PROGRAM 2**

**Design, develop and implement 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.**

**Source Code:**

#include<stdio.h>

void main()

{

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

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

printf("\nEnter the MAIN string: \n");

gets(STR); //Read the main string

printf("\nEnter a PATTERN string: \n");

gets(PAT); //Read the pattern string

printf("\nEnter a REPLACE string: \n");

gets(REP); //Read the replace string

i = m = c = j = 0;

while ( STR[c] != '\0') //Check till the end of the main string

{

// Checking for Match

if ( STR[m] == PAT[i] ) //Cmp main with pattern

{

i++;

m++;

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

{

//copy replace string in ans string

flag=1;

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

ans[j] = REP[k];

i=0;

c=m;

}

}

else //mismatch

{

ans[j] = STR[c]; //copy non-matched char to ans

j++;

c++;

m = c;

i=0;

}

}

if(flag==0)

{

printf("Pattern doesn't found!!!");

}

else

{

ans[j] = '\0';

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

}

}

**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)**

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.**

**Source Code:**

#include<stdlib.h>

#include<stdio.h>

#define max\_size 5

int stack[max\_size],top=-1;

/\*Function Prototype\*/

void push();

void pop();

void display();

void pali();

int main()

{

int choice;

while(1)

{

printf("\n\n--------STACK OPERATIONS-----------\n");

printf("1.Push\n");

printf("2.Pop\n");

printf("3.Palindrome\n");

printf("4.Display\n");

printf("5.Exit\n");

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

printf("\nEnter your choice:\t");

scanf("%d",&choice);

switch(choice)

{

case 1: push(); //to insert an item into stack

break;

case 2: pop(); //to delete an item from stack

break;

case 3: pali(); //to check palindrome or not

break;

case 4: display(); //to display items in stack

break;

case 5: exit(0); //exit the program

break;

default: printf("\n Invalid choice:\n");

break;

}

}

return 0;

}

void push() //Inserting element into the stack

{

int item,n;

if(top==(max\_size-1))

{

printf("\nStack Overflow:");

}

else

{

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

scanf("%d",&item);

top=top+1;

stack[top]=item;

}

}

void pop() //deleting an element from the stack

{

int item;

if(top==-1)

{

printf("Stack Underflow:");

}

else

{

item=stack[top];

top=top-1;

printf("\nThe poped element: %d\t",item);

}

}

void pali() //checking whether palindrome or not

{

int num[10],rev[10],i=0,k,flag=1;

k=top;

while(k!=-1)

{

num[i++]=stack[k--];

}

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

{

if(num[i]==stack[i])

continue;

else

flag=0;

}

if(top!=-1)

{

if(flag)

printf("It is palindrome number\n");

else

printf("It is not a palindrome number\n");

}

else

printf("Stack is Empty:");

}

void display() //Displaying the elements of stack

{

int i;

if(top==-1)

{

printf("\nStack is Empty:");

}

else

{

printf("\nThe stack elements are:\n" );

for(i=top;i>=0;i--)

{

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

}

}

}

**PROGRAM 4**

**Design, develop and implement 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.**

**Source Code:**

#include <ctype.h>

#include <stdio.h>

#define SIZE 50 /\* Size of Stack \*/

char s[SIZE]; /\* Global declarations \*/

int top = -1;

push(char elem) /\* Function for PUSH operation \*/

{

s[++top] = elem;

}

char pop() /\* Function for POP operation \*/

{

return (s[top--]);

}

int pr(char elem) /\* Function for precedence \*/

{

switch (elem)

{

case '#': return 0;

case '(': return 1;

case '+':

case '-': return 2;

case '\*':

case '/':

case '%': return 3;

case '^': return 4;

}

}

void main() /\* Main Program \*/

{

char infx[50], pofx[50], ch, elem;

int i = 0, k = 0;

printf("\n\n enter the Infix Expression : ");

gets(infx);

push('#');

while ((ch = infx[i++]) != '\0')

{

if (ch == '(')

push(ch);

else if (isalnum(ch))

pofx[k++] = ch;

else if (ch == ')')

{

while (s[top] != '(')

pofx[k++] = pop();

elem = pop(); /\* Remove ( \*/

}

else /\* Operator \*/

{

while (pr(s[top]) >= pr(ch))

pofx[k++] = pop();

push(ch);

}

}

while (s[top] != '#') /\* Pop from stack till empty \*/

pofx[k++] = pop();

pofx[k] = '\0'; /\* Make pofx as valid string \*/

printf("\n\n Given Infix Expn is: %s\n The Postfix Expn is:

%s\n", infx, pofx);

}

**PROGRAM 5**

**Design, develop and implement a Program in C for the following Stack Applications**

**Evaluation of Suffix expression with single digit operands and operators: +, -, \*, /,%, ^**

**Source Code :**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<string.h>

int s[30],op1,op2;

int top=-1,i;

char p[30],sym;

int op(int op1,char sym,int op2)

{

switch(sym)

{

case '+':return op1+op2;

case '-':return op1-op2;

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

case '/':return op1/op2;

case '%':return op1%op2;

case '^':

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

}

return 0;

}

int main()

{

printf("\nEnter the valid postfix exp:");

scanf("%s",p);

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

{

sym=p[i];

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

s[++top]=sym-'0';

else

{

op2=s[top--];

op1=s[top--];

s[++top]=op(op1,sym,op2);

}

}

printf("\nThe result is %d",s[top]);

getch();

}

**PROGRAM 6**

**Design, develop and implement a Program in C for the following Stack Applications**

**Solving Tower of Hanoi problem with n disks**

#include<stdio.h>

#include<conio.h>

int count=0,n;

int tower(int n,char s,char t,char d)

{

if(n==1)

{

printf("\n Move disc 1 from %c to %c",s,d);

count++;

return 1;

}

tower(n-1,s,d,t);

printf("\n Move disc %d from %c to %c",n,s,d);

count++;

tower(n-1,t,s,d);

}

int main( )

{

printf("\n Enter the no. of discs:");

scanf("%d",&n);

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

printf("\n The no. of disc moves is:%d",count);

getch( );

}

**PROGRAM 7**

**Design, develop and implement 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.**

**Source Code:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#define SIZE 5

int q[SIZE],i,r=-1,f=0,option,count=0,j;

int main( )

{

for(;;)

{

printf("\n1.Insert 2.Delete\n 3.Display 4.Exit");

printf("\nEnter your option:");

scanf("%d",&option);

switch(option)

{

case 1: //Inserting items to Queue

if(count==SIZE)printf("\n Q is Full\n");

else

{

r=(r+1)%SIZE;

printf("\nEnter the item:");

scanf("%d",&q[r]);

count++;

}

break;

case 2: //Deleting items from Queue

if(count==0)printf("\nQ is empty\n");

else

{

printf("\nDeleted item is: %d",q[f]);

count--;

f=(f+1)%SIZE;

}

break;

case 3: //Displaying items from Queue

if(count==0)printf("\nQ is Empty\n");

else

{

i=f;

for(j=0;j<count;j++)

{

printf(" %d",q[i]);

i=(i+1)%SIZE;

}

}

break;

default: exit(0);

}

}

}

**PROGRAM 8**

**Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: *USN, Name, Branch, 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 and Deletion at End of SLL**
4. **Perform Insertion and Deletion at Front of SLL**
5. **Demonstrate how this SLL can be used as STACK and QUEUE**
6. **Exit**

**Source Code:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct student

{

char usn[11];

char name[25];

int sem;

char branch[5];

int phno;

};

typedef struct student STUD;

struct node

{ char usn[11];

char name[25];

int sem;

char branch[5];

int phno;

struct node \*next;

};

typedef struct node NODE;

NODE \*first;

NODE\* copyNode(STUD s)

{

NODE \* temp;

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

if(temp==NULL)

{ printf("Memory cannot be allocated\n");

}

else

{

strcpy(temp->usn,s.usn);

strcpy(temp->name, s.name);

strcpy(temp->branch, s.branch);

temp->sem=s.sem;

temp->phno=s.phno;

temp->next=NULL;

return temp;

}

}

void addrear(STUD s) //Adding element at Rear End

{ NODE \*temp,\*cur;

temp=copyNode(s) ;

if(first==NULL)

{ temp=first; return;

}

cur=first;

while(cur->next != NULL)

{ cur=cur->next;

}

cur->next =temp;

return ;

}

void addfront(STUD s) //Adding element at Front End

{ NODE \*temp;

temp=copyNode(s); //ssn,name, dept, design,salary, pno);

if (first== NULL)

{

first=temp;

}

else

{ temp->next=first;

first=temp;

}

return ;

}

void display(NODE \*temp) //Displaying student info

{

printf("%s \t", temp->usn);

printf("%s \t", temp->name);

printf("%s \t", temp->branch);

printf("%d \t",temp->sem);

printf("%d \n", temp->phno);

}

void deletefront() //Deleting element from Front End

{

NODE \*temp; int num;

temp=first;

display(temp);

if(first==NULL)

{

printf("List is Empty"); return;

}

if(first->next==NULL)

first=NULL;

else

{ first=first->next;

}

free(temp);

return ;

}

void deleterear() //Deleting element from Rear End

{

NODE \*cur, \*prev;

cur=first;

prev=NULL;

if(first==NULL)

{ printf("List is Empty"); return;

}

if(first->next==NULL)

{ display(cur);

first=NULL;

free(cur); return;

}

while(cur->next!=NULL)

{ prev=cur;

cur=cur->next;

}

prev->next=NULL;

display(cur);

free(cur);

return;

}

void displayList() //Displaying all lists

{ NODE \*r;

r=first;

printf("USN\tName\tBrh\tSem\tPhone\n");

if(r==NULL)

return;

while(r!=NULL)

{ display(r);

r=r->next;

}

printf("\n");

}

STUD input() //Providing Input

{ STUD s;

printf("Enter USN : ");

scanf("%s",s.usn);

printf("Enter Name : ");

scanf("%s",s.name);

printf("Enter Branch: ");

scanf("%s",s.branch);

printf("Enter Sem:");

scanf("%d",&s.sem);

printf("Enter Phone no : ");

scanf("%d",&s.phno);

return s;

}

int count() //Counting number of nodes

{

NODE \*n;

int c=0;

n=first;

while(n!=NULL)

{ n=n->next;

c++;

}

return c;

}

int main()

{ STUD s; int i, ch;

first=NULL;

while(1)

{

printf("\nList Operations\n");

printf("===============\n");

printf("1.Create List by using front Insert\n");

printf("2.Display the status and count the nodes\n");

printf("3.Perform Insertion and Deletion at End of SLL\n");

printf("4.Perform Insertion and Deletion at Front of SLL\n");

printf("5.Demonstrate how this SLL can be used as STACK\n");

printf("6.Demonstrate how this SLL can be used as QUEUE\n");

printf("7.Exit\n");

printf("Enter your choice : ");

scanf("%d",&i);

switch(i)

{

case 1 : s=input();

addfront(s);

break;

case 2 : if(first==NULL)

{

printf("List is Empty\n");

}

else

{

printf(" Node Count=%d\t & Elements in the list are : \n", count());

displayList();

}

break;

case 3 : printf(" 1. Insert at End and 2 Delete From End=");

scanf("%d",&ch);

if(ch==1)

{ s=input();

addrear(s);

}

else if(ch==2)

deleterear();

else

printf(" Sorry wrong operation\n");

break;

case 4 : printf(" 1. Insert at Front and 2 Delete From Front=");

scanf("%d",&ch);

if(ch==1)

{ s=input();

addfront(s);

}

else if(ch==2)

deletefront();

else

printf(" Sorry wrong operation\n"); break;

case 5 : printf("This SLL can be used as STACK by Inserting at Rear end & Deleting from Rear end");

break;

case 6 : printf(" This SLL can be used as QUEUE by Inserting at Rear end & Deleting from Front end");

break;

case 7 : exit (0);

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

}

}

return 0;

}

**PROGRAM 9**

**Design, develop and implement 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**

**Source Code:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct employee

{

char ssn[11];

char name[21];

char dept[15];

char design[15];

int salary;

int phno;

};

typedef struct employee EMP;

struct node

{ char ssn[11];

char name[21];

char dept[15];

char design[15];

int salary;

int phno;

struct node \*next;

struct node \*prev;

};

typedef struct node NODE;

NODE \*first;

NODE\* copyNode(EMP e)

{ NODE \* temp;

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

if(temp==NULL)

{

printf("Memory cannot be allocated\n");

}

else

{ strcpy(temp->ssn,e.ssn);

strcpy(temp->name, e.name);

strcpy(temp->dept, e.dept);

strcpy(temp->design, e.design);

temp->salary=e.salary;

temp->phno=e.phno;

temp->next=NULL;

temp->prev=NULL;

return temp;

}

}

void addrear(EMP e) //Adding element at Rear End

{ NODE \*temp,\*cur, \*prev;

temp=copyNode(e) ;

if(first==NULL)

{ first=temp; return;

}

cur=first; prev=NULL;

while(cur->next != NULL)

{ prev=cur;

cur=cur->next;

}

cur->next =temp;

temp->prev=prev;

return ;

}

void addfront(EMP e) //Adding element at Front End

{ NODE \*temp;

temp=copyNode(e);

if (first== NULL)

{

first=temp;

}

else

{ temp->next=first;

first->prev=temp;

first=temp;

}

return ;

}

void display(NODE \*r) //Displaying Student info

{ printf("%s\t", r->ssn);

printf("%s\t", r->name);

printf("%s\t", r->dept);

printf("%s\t",r->design);

printf("%d\t",r->salary);

printf("%d\n", r->phno);

}

void deletefront() //Deleting elements from Front end

{

NODE \*temp; int num;

temp=first;

if(first==NULL)

{

printf(" List is Empty \n");return ;

}

if(first->next==NULL)

first=NULL;

else

{ first=first->next;

first->prev=NULL;

}

printf("SSN\tName\tDept\tDesignation\tSalary\tPhone\n");

display(temp);

free(temp);

return ;

}

void deleterear() //Deleting elements from Front end

{

NODE \*cur, \*prev;

cur=first;

prev=NULL;

if(first==NULL)

{

printf(" List is Empty \n");return ;

}

if(first->next==NULL)

{

first=NULL;

}

else

{

while(cur->next!=NULL)

{ prev=cur;

cur=cur->next;

}

prev->next=NULL;

}

printf("SSN\tName\tDept\tDesignation\tSalary\tPhone\n");

display(cur);

free(cur);

return;

}

void displayList() //Displaying all the nodes

{ NODE \*r;

r=first;

if(r==NULL)

{

return;

}

printf("SSN\t Name\t Dept\t Designation\t salary\tPhone\n");

while(r!=NULL)

{ display(r);

r=r->next;

}

printf("\n");

}

int count()

{

NODE \*n;

int c=0;

n=first;

while(n!=NULL)

{ n=n->next;

c++;

}

return c;

}

EMP input() //Providing Input

{ EMP e;

printf("Enter SSN : ");

scanf("%s",e.ssn);

printf("Enter Name : ");

scanf("%s",e.name);

printf("Enter dept: ");

scanf("%s",e.dept);

printf("Enter Disignation :");

scanf("%s",e.design);

printf("Enter Salary:");

scanf("%d",&e.salary);

printf("Enter Phone no : ");

scanf("%d",&e.phno);

return e;

}

int main()

{ EMP e; int i, ch;

first=NULL;

while(1)

{

printf("\nList Operations\n");

printf("===============\n");

printf("1.Create a DLL of N Employees Data by using end insertion\n");

printf("2.Display the status of DLL and count the number of nodes in it\n");

printf("3.Perform Insertion and Deletion at End of DLL\n");

printf("4.Perform Insertion and Deletion at Front of DLL\n");

printf("5.Demonstration of this DLL as Double Ended Queue\n");

printf("6.Exit\n");

printf("Enter your choice : ");

scanf("%d",&i);

switch(i)

{

case 1 : e=input();

addrear(e);

break;

case 2 : if(first==NULL)

{

printf("List is Empty\n");

}

else

{

printf(" Node Count=%d\t & Elements in the list are : \n", count());

displayList();

}

break;

case 3 : printf(" 1. Insert at End and 2 Delete From End=");

scanf("%d",&ch);

if(ch==1)

{ e=input();

addrear(e);

}

else if(ch==2)

deleterear();

else

printf(" Sorry wrong operation\n");

break;

case 4 : printf(" 1. Insert at Front and 2 Delete From Front=");

scanf("%d",&ch);

if(ch==1)

{ e=input();

addfront(e);

}

else if(ch==2)

deletefront();

else

printf(" Sorry wrong operation\n");

break;

case 5 : printf("This DLL can be used as Double Ended Queue by inserting and deleting from both ends \n");

break;

case 6 : return 0;

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

}

}

return 0;

}

**PROGRAM 10**

**Design, Develop and Implement a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes**

**Represent and Evaluate a Polynomial P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z-2xyz3**

**Source Code :**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

struct node //Defining Polynomial fields

{

int coef, px, py, pz,flag;

struct node \*link;

};

typedef struct node \* NODE;

NODE create\_list(NODE head) //For creating poly1 & poly2

{

int i,n,cf,px,py,pz;

printf("Enter the number of terms : ");

scanf("%d",&n);

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

{

printf("Enter the Co-ef, px, py, pz : ");

scanf("%d %d %d %d",&cf,&px,&py,&pz);

insert(head,cf,px,py,pz);

}

return head;

}/\*End of create\_list()\*/

insert(NODE head,int cof,int x,int y, int z) //inserting term to poly

{

NODE cur,tmp;

tmp= (NODE)malloc(sizeof(struct node)); //Allocates memory

int cf,px,py,pz;

cur=head->link;

tmp->coef=cof;

tmp->px=x;

tmp->py=y;

tmp->pz=z;

tmp->flag=0;

while(cur->link!=head)

{ cur=cur->link;

}

cur->link=tmp;

tmp->link=head;

}

int evaluate(NODE h, int x, int y, int z)

{ int v, s=0;

NODE cur=h->link;

while(cur!=h)

{

v=cur->coef\*pow(x, cur->px)\*pow(y, cur->py)\*pow(z,cur->pz);

s=s+v;

cur=cur->link;

}

return s;

}

void display(NODE head)

{

NODE cur;

if(head->link==head) //if poly is empty

{

printf("List is empty\n");

return;

}

cur=head->link;

while(cur != head) //display all terms till end

{

if(cur->coef > 0)

printf(" +%dx^%dy^%dz^%d ",cur->coef,cur->px,cur->py,cur->pz);

else if (cur->coef < 0)

printf(" %dx^%dy^%dz^%d ",cur->coef,cur->px,cur->py,cur->pz);

cur=cur->link;

}

printf("\n");

}/\*End of display() \*/

void main()

{

int choice,data,item,pos;

int x,y,z, value;

NODE head;

head=(NODE)malloc(sizeof(struct node));

head->link=head; //poly1

printf("\n1.Create Polynomial 1\n");

head=create\_list(head);

printf("\nPolynomial 1 is :");

display(head);

printf("Enter x, y and z for evaluation=");

scanf("%d%d%d", &x,&y,&z);

value=evaluate(head,x,y,z);

printf("\n Evaluation of Polynomial is :%d\n", value);

}

**PROGRAM 11**

**Design, Develop and Implement a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes**

**Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

struct node //Defining Polynomial fields

{

int coef, px, py, pz,flag;

struct node \*link;

};

typedef struct node \* NODE;

NODE create\_list(NODE head) //For creating poly1 & poly2

{

int i,n,cf,px,py,pz;

printf("Enter the number of terms : ");

scanf("%d",&n);

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

{

printf("Enter the Co-ef, px, py, pz : ");

scanf("%d %d %d %d",&cf,&px,&py,&pz);

insert(head,cf,px,py,pz);

}

return head;

}/\*End of create\_list()\*/

insert(NODE head,int cof,int x,int y, int z) //inserting term to poly

{

NODE cur,tmp;

tmp= (NODE)malloc(sizeof(struct node)); //Allocates memory

int cf,px,py,pz;

cur=head->link;

tmp->coef=cof;

tmp->px=x;

tmp->py=y;

tmp->pz=z;

tmp->flag=0;

while(cur->link!=head) //Identifying last node

cur=cur->link;

cur->link=tmp;

tmp->link=head;

}

NODE add\_poly(NODE h1,NODE h2,NODE h3)

{

NODE cur1,cur2,scf;

cur1=h1->link;

cur2=h2->link;

while(cur1 != h1) //Till end of poly1

{

if(cur2 == h2)

cur2=h2->link;

while(cur2 != h2) //Till end of poly2

{

if(cur1->px == cur2->px && cur1->py == cur2->py && cur1->pz == cur2->pz)

{ //Add & insert if co-ef's of both poly is equal

scf = cur1->coef + cur2->coef;

insert(h3,scf,cur1->px,cur1->py,cur1->pz);

cur2->flag=1;

cur2=h2->link;

break;

}

cur2=cur2->link;

}

if(cur1 == h1)

break;

if(cur2 == h2) //If co-ef of poly1 is not matched, insert it to poly3

insert(h3,cur1->coef,cur1->px,cur1->py,cur1->pz);

cur1=cur1->link;

}

cur2=h2->link;

while(cur2 != h2) //remaining poly2 nodes inserted to poly3

{

if(cur2->flag==0)

insert(h3,cur2->coef,cur2->px,cur2->py,cur2->pz);

cur2=cur2->link;

}

return h3;

}

void display(NODE head)

{

NODE cur;

if(head->link==head) //if poly is empty

{

printf("List is empty\n");

return;

}

cur=head->link;

while(c ur != head) //display all terms till end

{

if(cur->coef > 0)

printf(" +%dx^%dy^%dz^%d ",cur->coef,cur->px,cur->py,cur->pz);

else if (cur->coef < 0)

printf(" %dx^%dy^%dz^%d ",cur->coef,cur->px,cur->py,cur->pz);

cur=cur->link;

}

printf("\n");

}/\*End of display() \*/

void main()

{

int choice,data,item,pos;

NODE head1,head2,head3;

head1=(NODE)malloc(sizeof(struct node));

head1->link=head1; //poly1

head2=(NODE)malloc(sizeof(struct node));

head2->link=head2; //poly2

head3=(NODE)malloc(sizeof(struct node));

head3->link=head3; //poly3

printf("\n1.Create Polynomial 1\n");

head1=create\_list(head1);

printf("\n2.Create Polynomial 2\n");

head2=create\_list(head2);

printf("\nPolynomial 1 is :");

display(head1);

printf("\nPolynomial 2 is :");

display(head2);

head3=add\_poly(head1,head2,head3); //Add both polynomials

printf("\nAddition of two Polynomial is :");

display(head3);

}

**PROGRAM 12**

**Design, Develop and Implement 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 In-order, Preorder and Post Order**
3. **Search the BST for a given element (KEY) and report the appropriate message**
4. **Exit**

**Source Code:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int choice,data,key;

struct node //Defining Tree Structure

{

int info;

struct node \*lchild,\*rchild;

};

typedef struct node \*NODE;

NODE root=NULL;

NODE CREATE(NODE,int);

void INORDER(NODE);

void POSTORDER(NODE);

void PREORDER(NODE);

NODE DELETE\_NODE(NODE,int);

int main()

{

int choice, data;

while(1)

{

printf("\n1:CREATE\n”);

printf("\n 2:TREE TRAVERSAL\n 3.SEARCH\_AND\_DELETION\n4.EXIT");

printf("\n 3.SEARCH \n 4.EXIT");

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

scanf("%d",&choice);

switch(choice)

{

case 1: printf("\nEnter data to be inserted\n");

scanf("%d",&data);

root=CREATE(root,data);

break;

case 2: if(root==NULL)

printf("\nEMPTY TREE\n");

else

{

printf("\nThe Inorder display : ");

INORDER(root);

printf("\nThe Preorder display : ");

PREORDER(root);

printf("\nThe Postorder display : ");

POSTORDER(root);

}

break;

case 3: printf("\nenter the key to search and delete:\n");

scanf("%d",&key);

root=DELETE\_NODE(root,key);

break;

case 4: exit(0);

}

}

}

NODE CREATE(NODE root,int data) //Creating nodes of a tree

{

NODE newnode,x,parent;

newnode=(NODE)malloc(sizeof(struct node));

newnode->lchild=newnode->rchild=NULL;

newnode->info=data;

if(root==NULL)

root=newnode;

else

{

x=root;

while(x!=NULL)

{

parent=x;

if(x->info<data)

x=x->rchild;

else if(x->info>data)

x=x->lchild;

else

{

printf("\nNode is already present in the tree\n");

return(root);

}

}

if(parent->info<data)

parent->rchild=newnode;

else

parent->lchild=newnode;

}return(root);

}

void INORDER(NODE root) //In-order Traversal

{

if(root!=NULL)

{

INORDER(root->lchild);

printf("%d ",root->info);

INORDER(root->rchild);

}

}

void PREORDER(NODE root) //Pre-order Traversal

{

if(root!=NULL)

{

printf("%d ",root->info);

PREORDER(root->lchild);

PREORDER(root->rchild);

}

}

void POSTORDER(NODE root) //Post-order Traversal

{

if(root!=NULL)

{

POSTORDER(root->lchild);

POSTORDER(root->rchild);

printf("%d ",root->info);

}

}

**PROGRAM 13**

**Design, develop and implement 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 BFS method**
3. **Check whether a given graph is connected or not using DFS method.**

**Source Code:**

#include<stdio.h>

#include<stdlib.h>

int n,a[10][10],i,j,source,s[10],choice,count;

void bfs(int n,int a[10][10],int source,int s[]) //BFS Algorithm

{

int q[10],u;

int front=1,rear=1;

s[source]=1;

q[rear]=source;

while(front<=rear)

{

u=q[front];

front=front+1;

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

if(a[u][i]==1 &&s[i]==0)

{ rear=rear+1;

q[rear]=i;

s[i]=1;

}

}

}

void dfs(int n,int a[10][10],int source,int s[]) //DFS Algorithm

{

s[source]=1;

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

if(a[source][i]==1 && s[i]==0)

dfs(n,a,i,s);

}

int main()

{

printf("Enter the number of nodes : ");

scanf("%d",&n);

printf("\n Enter the adjacency matrix\n");

for(i=1;i<=n;i++) //Provide matrix of 0’s and 1’s

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

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

while(1)

{

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

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

scanf("%d",&choice);

switch(choice)

{

case 1: printf("\n Enter the source :");

scanf("%d",&source); //Provide source for BFS

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

s[i]=0;

bfs(n,a,source,s);

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

{

if(s[i]==0)

printf("\n The node %d is not reachable",i);

else

printf("\n The node %d is reachable",i);

}

break;

case 2:printf("\nEnter the source vertex :");

scanf("%d",&source); //Provide source for DFS

count=0;

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

s[i]=0;

dfs(n,a,source,s);

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

if(s[i])

count=count+1;

if(count==n)

printf("\nThe graph is connected.");

else

printf("\nThe graph is not connected.");

break;

case 3: exit(0);

}

}

}

**PROGRAM 14**

**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.**

**Design and 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.**

**Source Code:**

#include<stdio.h>

#include<conio.h>

#define MAX\_ADDR 5

struct employee

{

int emp\_id, emp\_age;

char emp\_name[25];

}emp[MAX\_ADDR];//Defining Array of Structures

void main()

{

int i, ch, count = 0, index, haddr, id, flag = 0;

for(;;)

{

printf("Enter 1 to insert record \n”);

printf(“Enter 2 to search record\n”);

printf(“Enter 3 to Exit\n");

scanf("%d", &ch);

switch(ch)

{

case 1: if(count == MAX\_ADDR) //When Records is full

{

printf("No free address space\n");

break;

}

printf("Enter employee id\n");

scanf("%d", &id);

haddr = hash(id);

printf("Home address is %d\n", haddr);

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

{

index = (haddr+i)%MAX\_ADDR;//For Wrap Around

if(emp[index].emp\_id == 0)

{

emp[index].emp\_id = id;

printf("Enter the employee name\n");

scanf("%s", emp[index].emp\_name);

printf("Enter the employee age\n");

scanf("%d", &emp[index].emp\_age);

count++;

printf("Inserted at Actual Addr%d:\n\n", index);

break;

}

}

break;

case 2: printf("Enter employee id to be searched\n");

scanf("%d",&id);

haddr = hash(id);

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

{

index = (haddr+i)%MAX\_ADDR;

if(emp[index].emp\_id == 0)

{

flag = 1;

break;

}

else if(emp[index].emp\_id == id)

{

printf("Employee id is%d\n", emp[index].emp\_id);

printf("Employee name is %s\n", emp[index].emp\_name);

printf("Employee age is %d\n", emp[index].emp\_age);

printf("Search Length is: %d\n", ++i);

break;

}

}

if(flag == 1 || i == MAX\_ADDR)

{

printf("Key not present\n");

}

break;

case 3: exit(0);

default: printf("Invalid Choice..\n");

}

}

}

int hash (int key) //Generates Hash address or Home address

{

return key % MAX\_ADDR;

}