**Program 01:**

#include <stdio.h>

#include <conio.h>

typedef struct distance

{

int kms;

int metres;

} DISTANCE;

DISTANCE add\_distance (DISTANCE, DISTANCE);

DISTANCE subtract\_distance(DISTANCE,DISTANCE);

DISTANCE dl, d2, d3, d4;

int main()

{

int option;

clrscr ();

do

{

printf("\n \*\*\*MAIN MENU\*\*\*");

printf ("\n 1. Read the distances ");

printf ("\n 2. Display the distances");

printf ("\n 3. Add the distances ");

printf ("\n 4. Subtract the distances");

printf ("\n 5. EXIT");

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

scanf("%d", &option);

switch(option)

{

case 1:

printf("\n Enter the first distance in kms and metres: ");

scanf ("%d %d", &dl .kms, &dl .metres);

printf("\n Enter the second distancekms and metres: ");

scanf ("%d %d" , &d2 .kms, &d2 .metres);

break;

case 2:

printf("\n The first distance is: %d kms %d metres " , dl.kms, dl.metres);

printf("\n The second distance is: %d kms %d metres " , d2 .kms, d2 .metres);

break;

case 3:

d3 = add\_distance(dl, d2);

printf("\n The sum of two distances is: %d kms %d metres", d3.kms, d3.metres);

break;

case 4:

d4 = subtract\_distance(dl, d2);

printf("\n The difference between two distances is: %d kms %d metres ", d4.kms, d4 .metres);

break;

}

}

while(option != 5);

{

getch ();

return 0;

}

}

DISTANCE add\_distance(DISTANCE dl, DISTANCE d2)

{

DISTANCE sum;

sum.metres = dl.metres + d2. metres;

sum.kms = dl.kms + d2.kms;

if(sum.metres >= 1000)

{

sum.metres = sum.metres%1000;

sum.kms += 1;

}

return sum;

}

DISTANCE subtract\_distance(DISTANCE dl,DISTANCE d2)

{

DISTANCE sub;

if(dl.kms > d2.kms)

{

sub.metres = dl.metres - d2. metres;

sub.kms = dl.kms - d2.kms;

}

else

{

sub.metres = d2.metres - dl. metres;

sub.kms = d2.kms - dl.kms;

}

if(sub.metres < 0)

{

sub.kms = sub.kms - 1;

sub.metres = sub.metres + 1000;

}

return sub;

}

**Program 02:**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<malloc.h>

struct node

{

int data;

struct node \*next;

};

struct node \*start=NULL;

struct node \*create\_ll(struct node \*);

struct node \*display(struct node \*);

struct node \*insert\_beg(struct node \*);

struct node \*insert\_end(struct node \*);

struct node \*delete\_node(struct node \*);

int main()

{

int option;

do

{

printf("\n\n \*\*\*\*\*\*\*\* MAIN MENU\*\*\*\*\*\*\*");

printf("\n 1: Create a list");

printf("\n 2: Display the list");

printf("\n 3: Insert beginning");

printf("\n 4: Insert end");

printf("\n 5: Delete A specified node");

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

scanf("%d",&option);

switch(option)

{

case 1: start=create\_ll(start);

printf("\nLinked list created");

break;

case 2: start=display(start);

break;

case 3: start=insert\_beg(start);

break;

case 4: start=insert\_end(start);

break;

case 5: start=delete\_node(start);

break;

}

}while(option!=6);

getch();

return 0;

}

struct node \*create\_ll(struct node \*start)

{

struct node \*new\_node,\*ptr;

int num;

printf("\n Enter -1 to end");

printf("\n Enter the data:");

scanf("%d",&num);

while(num !=-1)

{

new\_node=(struct node\*) malloc(sizeof(struct node));

new\_node->data=num;

if(start==NULL)

{

new\_node->next=NULL;

start=new\_node;

}

else

{

ptr=start;

while(ptr->next !=NULL)

ptr=ptr->next;

ptr->next=new\_node;

new\_node->next=NULL;

}

printf("Enter the data :");

scanf("%d",&num);

}

return start;

}

struct node \*display(struct node \*start)

{

struct node \*ptr;

ptr=start;

if(ptr==NULL)

{

printf("\nList is empty");

return 0;

}

while(ptr !=NULL)

{

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

ptr=ptr->next;

}

return start;

}

struct node \*insert\_beg(struct node \*start)

{

struct node \*new\_node;

int num;

printf("\n Enter the data");

scanf("%d",&num);

new\_node=(struct node \*)malloc(sizeof(struct node));

new\_node->data=num;

new\_node->next=start;

start=new\_node;

return start;

}

struct node \*insert\_end(struct node \*start)

{

struct node \*ptr, \*new\_node;

int num;

printf("\n Enter the data");

scanf("%d",&num);

new\_node=(struct node \*)malloc(sizeof(struct node));

new\_node->data=num;

new\_node->next=NULL;

ptr=start;

while(ptr->next !=NULL)

{

ptr=ptr->next;

}

ptr->next=new\_node;

return start;

}

struct node \*delete\_node(struct node \*start)

{

struct node \*ptr,\*preptr;

int val;

printf("\n Enter the value of the node which has to be deleted:");

scanf("%d",&val);

ptr=start;

if(ptr->data ==val)

{

start=start->next;

free(ptr);

return start;

}

else

{

while(ptr->data !=val)

{

preptr=ptr;

ptr=ptr->next;

}

preptr->next=ptr->next;

free(ptr);

return start;

}

}

**Program 03:**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<malloc.h>

#include<string.h>

struct node

{

struct node \*prev;

int book\_id;

char book\_title[10];

char author[20];

int edition;

struct node \*next;

};

struct node \*start=NULL;

struct node \*create\_ll(struct node \*);

struct node \*display(struct node \*);

struct node \*count\_node(struct node \*);

struct node \*delete\_pos(struct node \*);

int main()

{

int option;

clrscr();

do

{

printf("\n\n \*\*\*\*\*\*\*\* MAIN MENU\*\*\*\*\*\*\*");

printf("\n 1: Create a list");

printf("\n 2: Display the list");

printf("\n 3: Count nodes");

printf("\n 4: Delete postion");

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

scanf("%d",&option);

switch(option)

{

case 1: start=create\_ll(start);

printf("\nDoubly Linked list created");

break;

case 2: start=display(start);

break;

case 3: start=count\_node(start);

break;

case 4: start=delete\_pos(start);

break;

}

}while(option!=5);

getch();

return 0;

}

struct node \*create\_ll(struct node \*start)

{

struct node \*new\_node,\*ptr,\*preptr;

int book\_id,edition;

char book\_title[10],author[20];

printf("\n EnterBokk id as -1 to end");

printf("\n Enter the Book Id:");

scanf("%d",&book\_id);

printf("\n Enter the Book Title:");

scanf("%s",book\_title);

printf("\n Enter the Author:");

scanf("%s",author);

printf("\n Enter the Book Edition:");

scanf("%d",&edition);

while(book\_id!=-1)

{

new\_node=(struct node\*) malloc(sizeof(struct node));

new\_node->prev=NULL;

new\_node->book\_id=book\_id;

strcpy(new\_node->book\_title,book\_title);

strcpy(new\_node->author,author);

new\_node->edition=edition;

new\_node->next=NULL;

if(start==NULL)

{

start=new\_node;

}

else

{

ptr=start;

preptr=NULL;

while(ptr!=NULL&& book\_id> ptr->book\_id)

{

preptr=ptr;

ptr=ptr->next;

}

preptr->next=new\_node;

new\_node->next=ptr;

}

printf("Enter the Bookid:");

scanf("%d",&book\_id);

if(book\_id==-1)

break;

printf("\n Enter the Book Title:");

scanf("%s",book\_title);

printf("\n Enter the Author:");

scanf("%s",author);

printf("\n Enter the Book Edition:");

scanf("%d",&edition);

}

return start;

}

struct node \*display(struct node \*start)

{

struct node \*ptr;

ptr=start;

if(ptr==NULL)

{

printf("\nList is empty");

return 0;

}

while(ptr !=NULL)

{

printf("\t %d",ptr->book\_id);

printf("\t %s",ptr->book\_title);

printf("\t %s",ptr->author);

printf("\t %d",ptr->edition);

ptr=ptr->next;

}

return start;

}

struct node \*count\_node(struct node \*start)

{

struct node \*ptr;

int count=0;

ptr=start;

while(ptr!=NULL)

{

ptr=ptr->next;

count++;

}

printf("\nTotal Numbers of node=%d",count);

return start;

}

struct node \*delete\_pos(struct node \*start)

{

struct node \*ptr,\*preptr,\*temp;

int i,pos;

printf("Enter the postion");

scanf("%d",&pos);

temp=start;

for(i=1;i<pos && temp !=NULL;i++)

{

preptr=temp;

temp=temp->next;

ptr=temp->next;

}

if(temp!=NULL)

{

preptr->next=ptr;

ptr->prev=preptr;

free(temp);

}

else

{

printf("\nEnter valid postion");

}

return start;

}

**Program 04:**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<malloc.h>

#include<math.h>

struct node

{

int num;

int coeff;

struct node \*next;

};

struct node \*start1=NULL;

struct node \*start2=NULL;

struct node \*start3=NULL;

struct node \*start4=NULL;

struct node \*create\_poly(struct node \*);

struct node \*display\_poly(struct node \*);

struct node \*add\_poly(struct node \*,struct node \*,struct node \*);

struct node \*add\_node(struct node \*,int,int);

void evaluate(struct node \*);

int main()

{

int option;

clrscr();

do

{

printf("\n\n \*\*\*\*\*\*\*\* MAIN MENU\*\*\*\*\*\*\*");

printf("\n 1: Enter the First Polynomial");

printf("\n 2: Display the First Polynomial");

printf("\n 3: Enter the Second Polynomial ");

printf("\n 4: Display the Second Polynomial");

printf("\n 5: Add the polnomials");

printf("\n 6:Display the Result");

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

scanf("%d",&option);

switch(option)

{

case 1: start1=create\_poly(start1);

break;

case 2: start1=display\_poly(start1);

break;

case 3: start2=create\_poly(start2);

break;

case 4: start2=display\_poly(start2);

break;

case 5: start3=add\_poly(start1,start2,start3);

break;

case 6: start3=display\_poly(start3);

break;

}

}while(option!=7);

getch();

return 0;

}

struct node \*create\_poly(struct node \*start)

{

struct node \*new\_node,\*ptr;

int n,c;

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

scanf("%d",&n);

printf("\t Enter its coefficient:");

scanf("%d",&c);

while(n !=-1)

{

if(start==NULL)

{

new\_node=(struct node\*) malloc(sizeof(struct node));

new\_node->num=n;

new\_node->coeff=c;

new\_node->next=NULL;

start=new\_node;

}

else

{

ptr=start;

while(ptr->next !=NULL)

ptr=ptr->next;

new\_node=(struct node\*) malloc(sizeof(struct node));

new\_node->num=n;

new\_node->coeff=c;

ptr->next=new\_node;

new\_node->next=NULL;

}

printf("Enter the number :");

scanf("%d",&n);

if(n==-1)

break;

printf("\t Enter its Coefficient");

scanf("%d",&c);

}

return start;

}

struct node \*display\_poly(struct node \*start)

{

struct node \*ptr;

ptr=start;

while(ptr !=NULL)

{

printf("\n %d x %d\t",ptr->num,ptr->coeff);

ptr=ptr->next;

}

return start;

}

struct node \*add\_poly(struct node \*start1,struct node \*start2,struct node \*start3)

{

struct node \*ptr1,\*ptr2;

int sum\_num,c;

ptr1=start1,ptr2=start2;

while(ptr1 !=NULL && ptr2 !=NULL)

{

if(ptr1->coeff ==ptr2->coeff)

{

sum\_num=ptr1->num+ptr2->num;

start3=add\_node(start3,sum\_num,ptr1->coeff);

ptr1=ptr1->next;

ptr2=ptr2->next;

}

else if (ptr1->coeff > ptr2->coeff)

{

start3=add\_node(start3,ptr1->num,ptr1->coeff);

ptr1=ptr1->next;

}

else if (ptr1->coeff < ptr2->coeff)

{

start3=add\_node(start3,ptr2->num,ptr2->coeff);

ptr2=ptr2->next;

}

}

if(ptr1 ==NULL)

{

while(ptr2!=NULL)

{

start3=add\_node(start3,ptr2->num,ptr2->coeff);

ptr2=ptr2->next;

}

}

if(ptr2 ==NULL)

{

while(ptr1!=NULL)

{

start3=add\_node(start3,ptr1->num,ptr1->coeff);

ptr1=ptr1->next;

}

}

return start3;

}

struct node \*add\_node(struct node \*start,int n,int c)

{

struct node \*ptr,\*new\_node;

if(start==NULL)

{

new\_node=(struct node\*) malloc(sizeof(struct node));

new\_node->num=n;

new\_node->coeff=c;

new\_node->next=NULL;

start=new\_node;

}

else

{

ptr=start;

while(ptr->next !=NULL)

ptr=ptr->next;

new\_node=(struct node\*) malloc(sizeof(struct node));

new\_node->num=n;

new\_node->coeff=c;

ptr->next=new\_node;

new\_node->next=NULL;

}

return start;

}

**Program 05:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#define MAX 5

int s[MAX],top=-1;

void push(int s[],int val);

int pop(int s[]);

int peek(int s[]);

void display(int s[]);

int main()

{

int val,option;

do

{

printf("\n\*\*\*\*\*\* MAIN MENU\*\*\*\*\*\*\*\*");

printf("\n 1: PUSH");

printf("\n 2: POP");

printf("\n 3: PEEK");

printf("\n 4: DISPLAY");

printf("\n 5: EXIT");

printf("\n Enter Your option :");

scanf("%d",&option);

switch(option)

{

case 1: printf("\n Enter the Number to be pushed on stack");

scanf("%d",&val);

push(s,val);

break;

case 2:

val=pop(s);

if(val !=-1)

printf("\n The value deleted from stack is : %d",val);

break;

case 3:

val=peek(s);

if(val !=-1)

printf("\n The value stored at top of stack is : %d",val);

break;

case 4: display(s);

break;

}

}while(option != 5);

return 0;

}

void push (int s[],int val)

{

if(top==MAX-1)

{

printf("\n STACK OVERFLOW");

}

else

{

top++;

s[top]=val;

}

}

int pop(int s[])

{

int val;

if(top == -1)

{

printf("\n STACK UNDERFLOW");

return -1;

}

else

{

val=s[top];

top--;

return val;

}

}

int peek(int s[])

{

if(top == -1)

{

printf("\n STACK UNDERFLOW");

return -1;

}

else

return (s[top]);

}

void display(int s[])

{

int i;

if(top == -1)

{

printf("\n STACK EMPTY");

}

else

{

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

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

}

}