**Assignment 4**

**1). Program to implement various operation on Link List.**

**Code:**

#include<stdio.h>

struct Node{

int info;

struct Node \*next;

};

struct Node \* start;

struct Node \* CreateNode(void )

{

struct Node \* ptr;

ptr=(struct Node\*)malloc(sizeof(struct Node));

return ptr;

}

int Input()

{

int item;

printf("Enter the element to be inserted.");

scanf("%d",&item);

return item;

}

void InsertionAtBeginning()

{

struct Node\* new;

int item;

item=Input();

new=CreateNode();

new->info=item;

if(start==NULL)

{

new->next=NULL;

start=new;

}

else

{

new->next=start;

start=new;

}

}

void InsertionAtEnd()

{

struct Node\* ptr,\*new;

int item;

item=Input();

new=CreateNode();

new->info=item;

ptr=start;

// printf("%d",ptr->info);

if(start==NULL)

{

new->next=NULL;

start=new;

}

else

{

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

new->next=NULL;

ptr->next=new;

}

}

void InsertionInSortedList()

{

struct Node\* ptr,\*new;

int item;

item=Input();

new=CreateNode();

new->info=item;

ptr=start;

if(start==NULL)

{

new->next=NULL;

start=new;

}

else if(ptr->next==NULL){

if(item <= ptr->info)

{

new->next=ptr;

start=new;

}else{

ptr->next=new;

new->next=NULL;

}

}

else{

if(item <= ptr->info)

{

new->next=ptr;

start=new;

}else{

while(ptr->next!=NULL)

{

if((item > ptr->info)&&(item < ptr->next->info))

{

new->next=ptr->next;

ptr->next=new;

}else{

ptr=ptr->next;

}

}

}

}

}

void DeletionItemFromSortedList()

{

}

struct Node \* Search(int item)

{

struct Node\* ptr,\*temp=NULL;

ptr=start;

while(ptr!=NULL)

{

if(ptr->info==item)

{

// printf("Item found %d",ptr->info);

return ptr;

}

ptr=ptr->next;

}

return temp;

}

struct Node \* SearchP(int item)

{

struct Node\* ptr,\*temp=NULL,\*ptrsp;

ptr=start;

while(ptr!=NULL)

{

if(ptr->info==item)

{

// printf("Item found %d",ptr->info);

return ptrsp;

}

ptrsp=ptr;

ptr=ptr->next;

}

return temp;

}

void InsertionBefore()

{

struct Node\* new,\*ptrsp,\*ptrsn;

int item,newitem;

item=Input();

new=CreateNode();

new->info=item;

//printf("%d",ptr->info);

if(start==NULL)

{

new->next=NULL;

start=new;

}else

{

printf("\nEnter the exiting item in the linklist\nbefore which you want to insert the new item:");

scanf("%d",&newitem);

ptrsn=Search(newitem);

ptrsp=SearchP(newitem);

if(ptrsp==NULL)

{

printf("\nItem Not Found\n");

}else if(ptrsp==start){

new->next=ptrsp;

start=new;

}else

{

new->next=ptrsn;

printf("\nNext %d\n",ptrsn->info);

printf("\nPrev %d\n",ptrsp->info);

ptrsp->next=new;

}

}

}

void InsertionAfter()

{

struct Node\* new,\*ptrs;

int item,newitem;

item=Input();

new=CreateNode();

new->info=item;

if(start==NULL)

{

new->next=NULL;

start=new;

}else

{

printf("\nEnter the exiting item in the linklist\nafter which you want to insert the new item:");

scanf("%d",&newitem);

ptrs=Search(newitem);

//ptrsn=SearchN(newitem);

if(ptrs==NULL)

{

printf("\nItem Not Found\n");

}else if(ptrs==start){

new->next=NULL;

ptrs->next=new;

}else if(ptrs->next==NULL)

{

new->next=NULL;

ptrs->next=new;

}else

{

new->next=ptrs->next;

ptrs->next=new;

}

}

}

void DeletionFront()

{

struct Node \*ptr;

ptr=start;

if(ptr==NULL)

{

printf("Can't delete ! Empty Linklist !");

}else if(ptr->next==NULL)

{

printf("%d deleted",ptr->info);

start=NULL;

}else{

printf("%d deleted",ptr->info);

start=start->next;

if(start==NULL)

printf("Empty");

}

}

void DeletionLast()

{

struct Node \*ptr,\*ptrs;

ptr=start;

if(ptr==NULL)

{

printf("Can't delete ! Empty Linklist !");

}else if(ptr->next==NULL)

{

printf("%d deleted",ptr->info);

start=NULL;

}else{

while(ptr->next!=NULL)

{

ptrs=ptr;

ptr=ptr->next;

}

ptrs->next=NULL;

printf("%d deleted",ptr->info);

//free(ptr);

}

}

void DeletionofGivenItem()

{

struct Node \*ptr,\*ptrs,\*ptrsp;

int item;

printf("Enter the item to be deleted:");

scanf("%d",&item);

ptr=start;

if(ptr==NULL)

{

printf("Can't delete ! Empty Linklist !");

}else if(ptr->next==NULL)

{

printf("%d deleted",ptr->info);

start=NULL;

}else{

ptrs=Search(item);

if(ptrs==start)

{

ptrs=Search(item);

printf("First item %d deleted",ptrs->info);

start=ptrs->next;

}else{

ptrsp=SearchP(item);

ptrsp->next=ptrs->next;

printf("%d deleted",ptrs->info);

}

//free(ptr);

}

}

void Display()

{

struct Node\* ptr;

ptr=start;

if(ptr!=NULL)

{

printf("Elements of LinkList:\t");

while(ptr->next!=NULL)

{

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

ptr=ptr->next;

};

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

}else

{

printf("Empty Linklist");

}

}

int main()

{

int ch;

printf("\n1). Insertion at Beginning");

printf("\n2).Insertion at End ");

printf("\n3).Insertion in sorted list ");

printf("\n4).Insertion before ");

printf("\n5).Insertion after ");

printf("\n6).DeletionFront");

printf("\n7).DeletionLast");

printf("\n8).Deletion Of Given Item");

printf("\n9).Deletion From Sorted List");

printf("\n10).Display");

printf("\n11).Exit");

while(1)

{

printf("\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:InsertionAtBeginning();

break;

case 2:InsertionAtEnd();

break;

case 3:InsertionInSortedList();

break;

case 4:InsertionBefore();

break;

case 5:InsertionAfter();

break;

case 6:DeletionFront();

break;

case 7:DeletionLast();

break;

case 8:DeletionofGivenItem();

break;

case 9:DeletionItemFromSortedList();

break;

case 10:Display();

break;

case 11:exit(0);

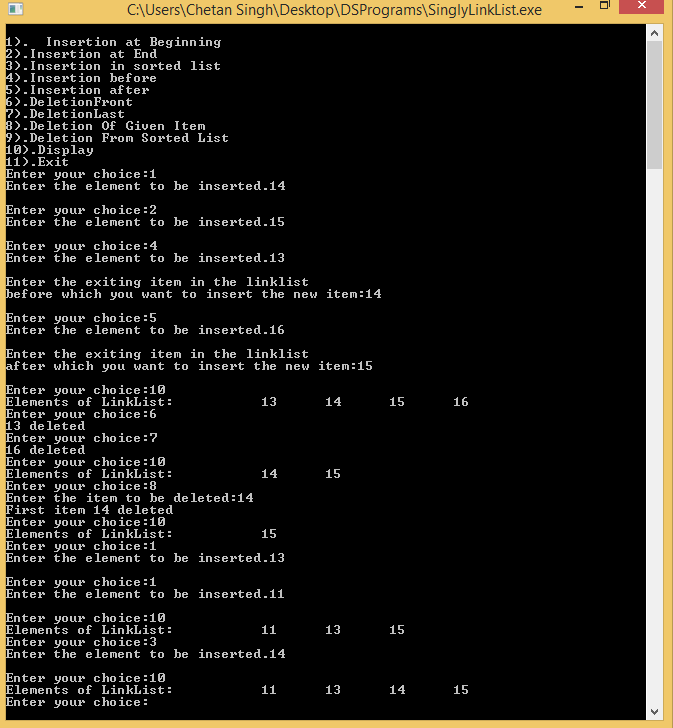
default:printf("Wrong choice.");

}

}

}

**Output:**



**1). Program to implement various operation on Link List.**

**Code:**

#include<stdio.h>

struct Node{

int info;

struct Node \*next;

struct Node \*prev;

};

struct Node \* start;

struct Node \* CreateNode(void )

{

struct Node \* ptr;

ptr=(struct Node\*)malloc(sizeof(struct Node));

return ptr;

}

int Input()

{

int item;

printf("Enter the element to be inserted.");

scanf("%d",&item);

return item;

}

void InsertionAtBeginning()

{

struct Node\* ptr;

int item;

item=Input();

ptr=CreateNode();

ptr->info=item;

//printf("%d",ptr->info);

if(start==NULL)

{

ptr->next=NULL;

ptr->prev=NULL;

start=ptr;

}

else

{

ptr->prev=ptr;

ptr->next=start;

start=ptr;

}

}

void InsertionAtEnd()

{

struct Node\* ptr,\*new;

int item;

item=Input();

new=CreateNode();

new->info=item;

ptr=start;

// printf("%d",ptr->info);

if(start==NULL)

{

new->next=NULL;

new->prev=NULL;

start=new;

}

else

{

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

new->prev=ptr;

new->next=NULL;

ptr->next=new;

}

}

void InsertionInSortedList()

{

}

struct Node \* Search(int item)

{

struct Node\* ptr,\*temp=NULL;

ptr=start;

while(ptr!=NULL)

{

if(ptr->info==item)

{

printf("Item found %d",ptr->info);

return ptr;

}

ptr=ptr->next;

}

return temp;

}

void InsertionBefore()

{

struct Node\* new,\*ptrs;

int item,newitem;

item=Input();

new=CreateNode();

new->info=item;

//printf("%d",ptr->info);

if(start==NULL)

{

new->next=NULL;

new->prev=NULL;

start=new;

}else

{

printf("\nEnter the exiting item in the linklist\nbefore which you want to insert the new item:");

scanf("%d",&newitem);

ptrs=Search(newitem);

if(ptrs==NULL)

{

printf("\nItem Not Found\n");

}else if(ptrs==start){

ptrs->prev=new;

new->next=ptrs;

new->prev=NULL;

start=new;

}else

{

new->next=ptrs;

new->prev=ptrs->prev;

ptrs->prev->next=new;

ptrs->prev=new;

}

}

}

void InsertionAfter()

{

struct Node\* new,\*ptrs;

int item,newitem;

item=Input();

new=CreateNode();

new->info=item;

//printf("%d",ptr->info);

if(start==NULL)

{

new->next=NULL;

new->prev=NULL;

start=new;

}else

{

printf("\nEnter the exiting item in the linklist\nafter which you want to insert the new item:");

scanf("%d",&newitem);

ptrs=Search(newitem);

if(ptrs==NULL)

{

printf("\nItem Not Found\n");

}else if(ptrs==start){

new->prev=ptrs;

new->next=NULL;

ptrs->next=new;

}else if(ptrs->next==NULL)

{

new->prev=ptrs;

new->next=NULL;

ptrs->next=new;

}else

{

new->prev=ptrs;

new->next=ptrs->next;

ptrs->next->prev=new;

ptrs->next=new;

}

}

}

void DeletionFront()

{

struct Node \*ptr;

ptr=start;

if(ptr==NULL)

{

printf("Can't delete ! Empty Linklist !");

}else if(ptr->next==NULL)

{

printf("%d deleted",ptr->info);

start=NULL;

}else{

printf("%d deleted",ptr->info);

start=start->next;

if(start==NULL)

printf("Empty");

}

}

void DeletionLast()

{

struct Node \*ptr;

ptr=start;

if(ptr==NULL)

{

printf("Can't delete ! Empty Linklist !");

}else if(ptr->next==NULL)

{

printf("%d deleted",ptr->info);

start=NULL;

}else{

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

printf("%d deleted",ptr->info);

ptr->prev->next=NULL;

free(ptr);

}

}

void Display()

{

struct Node\* ptr;

ptr=start;

if(ptr!=NULL)

{

printf("Elements of LinkList:\t");

while(ptr->next!=NULL)

{

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

ptr=ptr->next;

};

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

}else

{

printf("Empty Linklist");

}

}

int main()

{

int ch;

printf("\n1). Insertion at Beginning");

printf("\n2).Insertion at End ");

printf("\n3).Insertion in sorted list ");

printf("\n4).Insertion before ");

printf("\n5).Insertion after ");

printf("\n6).DeletionFront");

printf("\n7).DeletionLast");

printf("\n8).Display");

printf("\n9).Exit");

while(1)

{

printf("\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:InsertionAtBeginning();

break;

case 2:InsertionAtEnd();

break;

case 3:InsertionInSortedList();

break;

case 4:InsertionBefore();

break;

case 5:InsertionAfter();

break;

case 6:DeletionFront();

break;

case 7:DeletionLast();

break;

case 8:Display();

break;

case 9:exit(0);

default:printf("Wrong choice.");

}

}

}

Output:

