```
//Created By Ritwik Chandra Pandey on 28/02/2021
//183215
//Cursor Implementation of Linked List


#include<stdio.h>
#include<stdlib.h>


#define SPACE_SIZE 11
struct Node
{
   int data;
   int Next;
};
typedef int PtrToNode;
typedef PtrToNode POSITION;
typedef PtrToNode LIST;
struct Node CursorSpace[SPACE_SIZE];

void InitializeCursor()
{
   int i;
   for(i=0;i<SPACE_SIZE-1;i++)
   {
      CursorSpace[i].Next=i+1;
      CursorSpace[i].data=0;
   }
   CursorSpace[SPACE_SIZE-1].Next=0;
   CursorSpace[SPACE_SIZE-1].data=0;
}
POSITION CursorAlloc()
{
   POSITION P;
```

```
   P=CursorSpace[0].Next;
   CursorSpace[0].Next=CursorSpace[P].Next;
   CursorSpace[P].data = 0;
   CursorSpace[P].Next = 0;
   return P;
}
void CursorFree(POSITION P)
{
   CursorSpace[P].Next=CursorSpace[0].Next;
   CursorSpace[0].Next=P;
   CursorSpace[P].data = 0;

}
void Insert(int X,LIST L,POSITION P)// ElementType in general for x's data type
{/*Header Implementation assumed and Parameter L is unused in this implementation*/
   POSITION Temp;
   Temp=CursorAlloc();
   if(Temp==0)
      printf("\nOut of space!!!");
   else
   {

      CursorSpace[Temp].data=X;
      CursorSpace[Temp].Next=CursorSpace[P].Next;
      CursorSpace[P].Next=Temp;
   }
}
int IsLast(POSITION P, LIST L)  //P is assumed to be a valid position in List L
{
   return CursorSpace[P].Next==0;
}
int IsEmpty(LIST L)
{
   return CursorSpace[L].Next==0;
}
```

```
POSITION Find(int X,LIST L)
{
   POSITION Temp;
   Temp=CursorSpace[L].Next; //first node is header node
   while(Temp && CursorSpace[Temp].data!=X)
      Temp=CursorSpace[Temp].Next;
   return Temp;
}
POSITION FindPrevious(int X,LIST L)
{
   POSITION Temp;
   Temp=L;
   while(CursorSpace[Temp].Next && CursorSpace[CursorSpace[Temp].Next].data!=X)
   {Temp=CursorSpace[Temp].Next;}
   if(CursorSpace[Temp].Next==0){
     return 0;}
   else  return Temp;

}
void Delete(int X,LIST L)
{ //Assume use of a header node
   POSITION P,Temp;
   P=FindPrevious(X,L);


      Temp=CursorSpace[P].Next;
      CursorSpace[P].Next=CursorSpace[Temp].Next;
      CursorFree(Temp);

}
void DeleteList(LIST L)
{POSITION P, Temp;
   P = CursorSpace[L].Next;
   CursorSpace[L].Next = 0; //Header assumed
```

```
    while(P!=0)
    {Temp = CursorSpace[P].Next;
      CursorFree(P);
      P = Temp;}
}
void Display()
{
   int i;
   for(i=0;i<=SPACE_SIZE-1;i++)
      printf("\n%d\t%d\t%d",i,CursorSpace[i].data,CursorSpace[i].Next);
}



int main()
{
   LIST L=0;
   POSITION P;
   int choice,place,x;
   printf("\n\tCursor Implementation of List ADT\n\n");
   do{

      printf("\n1.Create\n2.Insert\n3.Delete\n4.DeleteList\n5.Display\n6.Find\n7.Exit\n\n");

      printf("\nEnter your choice:\t");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
            if(L==0)
            {
               InitializeCursor();

               L=CursorAlloc();
```

```
            printf("Cursor initialized and has been allocated.\n");
        }
        else
            printf("List has already been created\n");
        break;
    case 2:
        if(L==0)
            printf("\nList is not yet initialized");
        else
        {
            printf("\nWhere do you want to insert? (Please enter a valid position) ");
            scanf("%d",&place);
            printf("\nEnter the element to insert: ");
            scanf("%d",&x);
            Insert(x,L,place);
        }
        break;
    case 3:
        if(L==0)
            printf("List is not yet initialized\n");
        else
        {
            printf("Which element do you want to delete? ");
            scanf("%d",&x);
            Delete(x,L);
        }
        break;
    case 4:
        if(L==0)
            printf("\nList is not yet initialized");
        else
            DeleteList(L);
        break;
```

```
        case 5:
            if(L==0)
                printf("\nList is not yet initialized");
            else
                Display();
            break;
        case 6:
            if(L==0)
                printf("\nList is not yet initialized");
            else
            {
                printf("\nWhich element do you want to search? ");
                scanf("%d",&x);
                P=Find(x,L);
                printf("\nThe element is at %d",P);
            }
            break;
        case 7:
            exit(0);
        default:
            printf("\n Please enter correct option");
      }
   }while(choice!=7);
}
```