**ASSIGNMENT 2 > PROGRAM 1: LINKED LIST**

*Source Code –*

/\*Linked List

Author: Asmit De \*/

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

typedef struct node

{

int data;

struct node \*next;

}NODE;

NODE \*header=NULL;

NODE \*getNode(int data)

{

NODE \*node = (NODE\*)calloc(1,sizeof(NODE));

node->data = data;

node->next = NULL;

return node;

}

void returnNode(NODE \*node)

{

free(node);

}

NODE \*addBegin(NODE \*list,int data)

{

NODE \*node = getNode(data);

node->next = list;

return node;

}

int addLast(NODE \*list,int data)

{

NODE \*node;

if(list == NULL)

return 1;

while(list->next != NULL)

list = list->next;

node = getNode(data);

list->next = node;

return 0;

}

int addAfterPos(NODE \*list,int pos,int data)

{

NODE \*node;

int i=1;

if(list == NULL)

return 1;

while(i<pos)

{

if(list->next != NULL)

{

list = list->next;

i++;

}

else

return 1;

}

node = getNode(data);

node->next = list->next;

list->next = node;

return 0;

}

int addAfterData(NODE \*list,int edata,int data)

{

NODE \*node;

if(list == NULL)

return 1;

while(list != NULL && list->data != edata)

{

list = list->next;

if(list == NULL)

return 1;

}

node = getNode(data);

node->next = list->next;

list->next = node;

return 0;

}

NODE \*delBegin(NODE \*list)

{

NODE \*node = list;

list = list->next;

returnNode(node);

return list;

}

int delLast(NODE \*list)

{

NODE \*node;

if(list == NULL)

return 1;

else if(list->next == NULL)

{

header = delBegin(list);

return 0;

}

else

{

while((list->next)->next != NULL)

list = list->next;

}

node = list->next;

returnNode(node);

list->next = NULL;

return 0;

}

int delPos(NODE \*list,int pos)

{

NODE \*node;

int i=1;

if(list == NULL)

return 1;

else if(pos == 1)

{

header = delBegin(list);

return 0;

}

else

{

while(i<pos-1)

{

list = list->next;

i++;

if(list->next == NULL)

return 1;

}

}

node = list->next;

list->next = node->next;

returnNode(node);

return 0;

}

int delData(NODE \*list,int edata)

{

NODE \*node;

if(list == NULL)

return 1;

else if(list->data == edata)

{

header = delBegin(list);

return 0;

}

else if(list->next == NULL)

return 1;

else

{

while((list->next)->data != edata)

{

list = list->next;

if(list->next == NULL)

return 1;

}

}

node = list->next;

list->next = node->next;

returnNode(node);

return 0;

}

void displayList(NODE \*list)

{

printf("List: ");

if(list != NULL)

do

{

printf("%d->",list->data);

list = list->next;

}while(list != NULL);

printf("NULL");

}

NODE \*reverseList(NODE \*list,NODE \*rem)

{

NODE \*temp = NULL;

if(list == NULL)

return rem;

temp = list->next;

list->next = rem;

return reverseList(temp,list);

}

void insertSorted(NODE \*list, int data)

{

NODE \*node = NULL;

if(list == NULL)

header = getNode(data);

else

{

while(list->next!=NULL && list->next->data < data)

list= list->next;

node = getNode(data);

node->next = list->next;

list->next = node;

}

}

int extrema(NODE \*list)

{

int max, min, i=1; float sum=0, avg;

if(list == NULL)

return 1;

else

{

max = min = list->data;

while(list->next != NULL)

{

list = list->next;

i++;

sum += list->data;

if(list->data > max) max = list->data;

if(list->data < min) min = list->data;

}

avg = sum / i;

printf("Maximum = %d\nMinimum = %d\nAverage = %.2f", max, min, avg);

return 0;

}

}

int search(NODE \*list, int data)

{

while(list != NULL)

{

if(list->data == data)

return 0;

list = list->next;

}

return 1;

}

void main()

{

char ch;

int data,edata,pos,flag;

while(1)

{

system("cls");

puts("MENU");

puts("Key\tFunction");

puts("1\tDisplay list");

puts("2\tAdd node at beginning of list");

puts("3\tAdd node at end of list");

puts("4\tAdd node after a position in list");

puts("5\tAdd node after an existing data in list");

puts("6\tDelete node from beginning of list");

puts("7\tDelete node from end of list");

puts("8\tDelete node at a position in list");

puts("9\tDelete node containing an existing data in list");

puts("0\tReverse list");

puts("i\tInsert data in correct position in sorted list");

puts("e\tFind extrema and average");

puts("s\tSearch for a data in the list");

puts("x\tExit");

puts("\nEnter your choice...");

ch=getch();

flag=0;

switch(ch)

{

case '1':

system("cls");

displayList(header);

puts("\nPress any key to return to menu...");

getch();

break;

case '2':

system("cls");

printf("Enter data to add: ");

scanf("%d",&data);

header = addBegin(header,data);

printf("\nData added successfully...\nPress any key to return to menu...");

getch();

break;

case '3':

system("cls");

printf("Enter data to add: ");

scanf("%d",&data);

if(!addLast(header,data))

printf("\nData added successfully...\nPress any key to return to menu...");

else

printf("\nError: List is empty...\nPress any key to return to menu...");

getch();

break;

case '4':

system("cls");

printf("Enter data to add: ");

scanf("%d",&data);

printf("Enter the position after which the data is to be added: ");

scanf("%d",&pos);

if(!addAfterPos(header,pos,data))

printf("\nData added successfully...\nPress any key to return to menu...");

else

printf("\nError: Invalid position number...\nPress any key to return to menu...");

getch();

break;

case '5':

system("cls");

printf("Enter data to add: ");

scanf("%d",&data);

printf("Enter the existing data after which the data is to be added: ");

scanf("%d",&edata);

if(!addAfterData(header,edata,data))

printf("\nData added successfully...\nPress any key to return to menu...");

else

printf("\nError: Given data not found...\nPress any key to return to menu...");

getch();

break;

case '6':

system("cls");

if(header != NULL)

{

header = delBegin(header);

printf("\nNode deleted successfully...\nPress any key to return to menu...");

}

else

printf("\nError: List is empty...\nPress any key to return to menu...");

getch();

break;

case '7':

system("cls");

if(!delLast(header))

printf("\nNode deleted successfully...\nPress any key to return to menu...");

else

printf("\nError: List is empty...\nPress any key to return to menu...");

getch();

break;

case '8':

system("cls");

printf("Enter the position of node which is to be deleted: ");

scanf("%d",&pos);

if(!delPos(header,pos))

printf("\nNode deleted successfully...\nPress any key to return to menu...");

else

printf("\nError: Invalid position number...\nPress any key to return to menu...");

getch();

break;

case '9':

system("cls");

printf("Enter the data which is to be deleted: ");

scanf("%d",&edata);

if(!delData(header,edata))

printf("\nNode deleted successfully...\nPress any key to return to menu...");

else

printf("\nError: Given data not found...\nPress any key to return to menu...");

getch();

break;

case '0':

system("cls");

header = reverseList(header,NULL);

printf("\nList reversed successfully...\nPress any key to return to menu...");

getch();

break;

case 'i':

case 'I':

system("cls");;

printf("Enter data to add: ");

scanf("%d",&data);

insertSorted(header,data);

printf("\nData added successfully...\nPress any key to return to menu...");

getch();

break;

case 'e':

case 'E':

system("cls");

if(extrema(header))

printf("\nError: List is empty...\nPress any key to return to menu...");

getch();

break;

case 's':

case 'S':

system("cls");

printf("Enter data to search: ");

scanf("%d",&data);

if(!search(header, data))

printf("\n%d is present in list...\nPress any key to return to menu...");

else

printf("Given data not found...\nPress any key to return to menu...");

getch();

break;

case 'x':

flag=1;

break;

default:

system("cls");

puts("Invalid input...\nPress any key to return to menu...");

getch();

break;

}

if(flag)

break;

}

}