University of Engineering and Technology, Lahore (New Campus)

Department of Electrical Engineering and Technology

Data Structures and Algorithms

Lab 7: Linked lists

**Q1):** Write a program that implements the following:

Functions:

1. Create () adding the very first node.

void creating\_node(int value)

{

try

{

if (start == NULL)

{

start=new node;

start->data=value;

start->next=NULL;

length=length+1;

}

else

{

throw(value);

}

}

catch(int data)

{

cout<<"Linked list already exist, Please try to add instead of creating linked list"<<endl;

}

return;

}

1. Add() the method should insert numbers in sorted order.

void add(int value)

{

if (length==0)

{

cout<<"There is no sign of linked list, Try to create linked list first"<<endl;

}

else

{

node \*temp;

temp = start;

while (temp->next!=NULL)

{

temp=temp->next;

}

node \* new\_node;

new\_node = new node;

new\_node->data=value;

new\_node->next=NULL;

temp->next=new\_node;

length = length+1;

}

return;

}

1. Delete() the function should clearly show, with appropriate comments, deleting a

node (if it is present) by enumerating all the elements of the list after deleting the

concerned node, and “not found” if the requested node is not present in the list.

void deleting(int value)

{

try

{

if (length==0)

{

throw(1);

cout<<"Linked list is empty, deleting is not possible"<<endl;

}

else

{

node \*current, \*prev;

current = start;

while ((current->next!=NULL) && (current->data!= value))

{

prev = current;

current = current->next;

}

if (current->data==value)

{

// if it is first node then

if (current==start)

{

start = current->next;

}

else

{

prev->next=current->next;

}

delete current;

length=length-1;

}

}

cout<<"Noting to delete"<<endl;

}

catch(...)

{

cout<<"Node having value ["<<value<<"] not found, Hence nothing to delete"<<endl;

}

return;

}

1. Traverse() informs the user of the length of the linked list

int sized\_of\_linked\_list()

{

int size=1;

node \*temp = start;

while (temp->next!=NULL)

{

temp=temp->next;

size = size +1;

}

return size;

}

1. Traverse () simply prints the elements of the list.

void traverse()

{

try

{

if (length==0)

{

throw(1);

}

else

{

node \*temp;

temp = start;

int len=1;

while (temp->next!=NULL)

{

cout<<"["<<len<<"]: "<<temp->data<<" "<<endl;

temp=temp->next;

len = len +1;

}

cout<<"["<<len<<"]: "<<temp->data<<" "<<endl;

}

}

catch(...)

{

cout<<"Linked list is empty, Try create and add then traverse"<<endl;

}

return;

}

A main( ) function that:

1. Text calls upon an external file containing the numbers [1, 3, 7, 99, 101, 103,107] (you can create a text file yourself.
2. creates and populates the linked list with all the elements presented to it in the text file of part (2a)
3. thereafter, the function asks the user a series of actions that he/she would like to continue to perform iteratively up until he/she chooses to stop. Actions are:

(i) add a node

(ii) delete a node

(iii) enumerate the elements of the list

(iv) length of the list

(v) end – stop the process and stores the current linked list in an external file called

“output.txt”

**Code:**

int main()

{

ifstream inputFile("sli.txt");

if (inputFile.is\_open()) {

int num;

while (inputFile >> num) {

if (start == NULL) {

creating\_node(num);

} else {

add(num);

}

}

inputFile.close();

} else {

cout << "Unable to open numbers.txt" << endl;

return 1;

}

int choice;

do

{

cout<<endl;

cout << "Choose any single action:" << endl;

cout << "[1] Add a node?" << endl;

cout << "[2] Delete a node?" << endl;

cout << "[3] Enumerate the elements of the list?" << endl;

cout << "[4] Length of the list?" << endl;

cout << "[5] End and store the current linked list?" << endl;

cout<<"-----------------------------------------------------------------------------"<<endl;

cout << "Enter your choice: ";cin>>choice;cout<<endl;

switch (choice)

{

case 1:{

int input\_value;

cout<<"Enter the value of node: ";cin>>input\_value;cout<<endl;

add(input\_value);

break;

}

case 2:{

int input\_value;

cout<<"Enter the value of node that you want to delete: ";cin>>input\_value;cout<<endl;

deleting(input\_value);

break;

}

case 3:{

traverse();

break;

}

case 4:{

cout<<"Length of the single linked list: "<<size\_of\_linked\_list()<<endl;

break;

}

case 5:{

saveListToFile();

break;

}

default:

cout << "Invalid choice. Please try again." << endl;

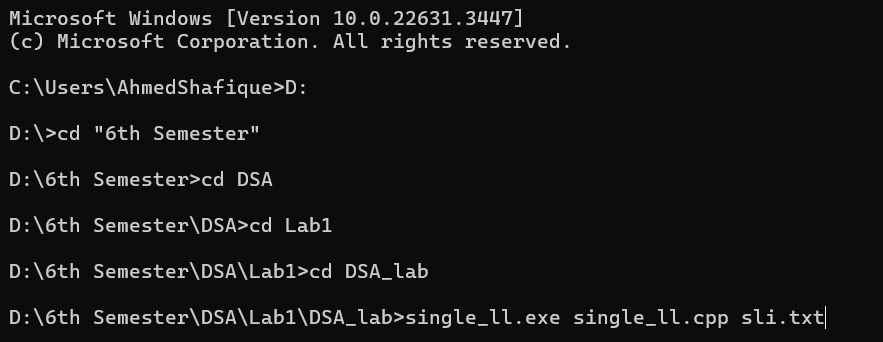
}

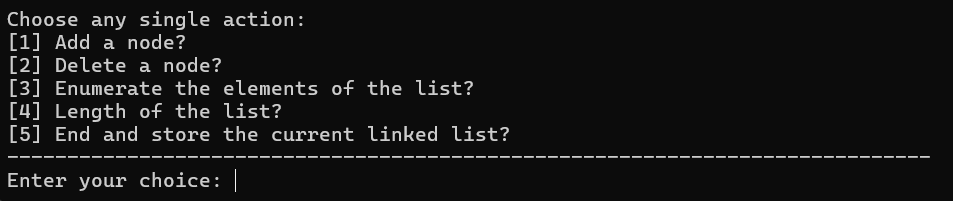
}while(choice!=5);

return 0;

}

Output:





A black background with white text

Description automatically generated

A black screen with white text

Description automatically generated

A computer screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

