Data Structures and Algorithms

Lab 7: Linked lists

Q1): Write a program that implements the following:

Functions:

a. 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;
}
```

b. 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;
}
```

c. 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;
}
```

d. 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;
}
```

e. 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:

a. Text calls upon an external file containing the numbers [1, 3, 7, 99, 101, 103,107] (you can create a text file yourself.

b. creates and populates the linked list with all the elements presented to it in the text file of part (2a)

c. 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:

Command Prompt:

```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\AhmedShafique>D:

D:\>cd "6th Semester"

D:\6th Semester>cd DSA

D:\6th Semester\DSA>cd Lab1

D:\6th Semester\DSA\Lab1>cd DSA_lab

D:\6th Semester\DSA\Lab1\DSA_lab>single_ll.exe single_ll.cpp sli.txt
```

Fig: 01 Programming running on CMD

```
Choose any single action:
[1] Add a node?
[2] Delete a node?
[3] Enumerate the elements of the list?
[4] Length of the list?
[5] End and store the current linked list?
--------------------------------------------------------------------------------
Enter your choice:
```

Fig: 02 Accepting Options

```
Enter your choice: 3

[1]: 1
[2]: 3
[3]: 9
[4]: 99
[5]: 101
[6]: 103
[7]: 107
```

Fig: 03 Traversing Linked list

```
Choose any single action:
[1] Add a node?
[2] Delete a node?
[3] Enumerate the elements of the list?
[4] Length of the list?
[5] End and store the current linked list?
--------------------------------------------------------------------------------
Enter your choice: 1

Enter the value of node: 48
```

Fig: 04 Adding node

```
Choose any single action:
[1] Add a node?
[2] Delete a node?
[3] Enumerate the elements of the list?
[4] Length of the list?
[5] End and store the current linked list?
--------------------------------------------------------------------------------
Enter your choice: 2

Enter the value of node that you want to delete: 48
```

Fig: 05 Deleting node

```
Choose any single action:
[1] Add a node?
[2] Delete a node?
[3] Enumerate the elements of the list?
[4] Length of the list?
[5] End and store the current linked list?
--------------------------------------------------------------------------------
Enter your choice: 4

Length of the single linked list: 7
```

Fig: 06 Linked lists length

```
Choose any single action:
[1] Add a node?
[2] Delete a node?
[3] Enumerate the elements of the list?
[4] Length of the list?
[5] End and store the current linked list?
--------------------------------------------------------------------------------
Enter your choice: 5

Linked list saved to output.txt
```

Fig: 07 Saving text file

Methodology:

First, we added required header files to run the program which are:

```
#include <iostream>
#include <new>
#include <fstream>
using namespcace std;
```

The according to the requirement of making structure of linked list we code the structure of linked list which are as follows:

```
struct node
{
    int data;
    node *next;
};
```

Then we initialize the linked list to zero because there is no sign of linked list initially using following code:

```
int length = 0;
node *start = NULL;
```

We declare them as global because now these can easily calls in any function so there is no need to additionally passed them into function as a argument. Then we coded the `creating_node` function to create the node as user input text file. Then we coded `traverse` function, the traverse function prints the value in whole linked list and we set that if there no node then it return nothing and we use `while` loop and pass the condition that it stops where the next structure pointer points to `NULL`.

We coded the `add` function to add additional nodes from the user and then, we coded another function called `deleting` to delete the node of linked list that user want to delete based on the value of the node input by the user.

We coded `size_of_linked_list` function to calculate the length of linked list, it same as some portion of `traverse` function but it have one statement different from traverse which is:

```
length = length + 1;
```

then it returning length using `return length;`

At last we coded the last part of program in which user can save the linked list data into the output text file. So, We initialize the file input in main function then we pass one condition that if there is no sign of linked list and it initialize to zero then it calls the `creating_node` function to create the node in the linked list and initialize it to `1`. `Else` it add the nodes in the linked list using `add` function.

Also we give choice to user to save it linked lists data that the user ran into output text file using `ofstream`, here is the code below:

```
void saveListToFile() {
    ofstream outputFile("slo.txt");
    if (outputFile.is_open()) {
        node *temp = start;
        while (temp != nullptr) {
            outputFile << temp->data << " ";
            temp = temp->next;
        }
        outputFile.close();
        cout << "Linked list saved to output.txt" << endl;
    } else {
        cout << "Unable to open output.txt for writing" << endl;
    }
}
```

Data Structures and Algorithms

Lab: 08 Double Linked lists

Q1. Write a program that implements the following:

Functions:

a) create(): adding the very first node

```
void create (int value) {
 if (length == 0) {
      start = new node;
   start->data = value;
   start->previous = NULL; // Use nullptr instead of NULL for
better clarity
   start->next = NULL;
   length = length + 1; // Don't forget semicolon at the end of the
statement
 }
 else {
     cout << "List not empty. Try Again." << endl; // Fixed the error
message string
 }
    return;
}
```

b) add (): the method should insert numbers in sorted order.

```
void add(int value) {
    node* newNode = new node;
    newNode->data = value;
    newNode->previous = NULL;
    newNode->next = start;
    if (start != NULL)
        start->previous = newNode;
    start = newNode;
    length = length + 1;
}
```

c) delete () : the function should clearly show, with appropriate comments, deleting a node (if it is present), and "not found" if the requested node is absent.

```
void deleteNode(int value) {
```

```
    if (length == 0) {
        cout << "List is empty." << endl;
        return;
    }

    node* current = start;
    node* previous = NULL;
    while (current != NULL) {
        if (current->data == value) {
            if (current == start) {
                start = current->next;
                if (start != NULL)
                    start->previous = NULL;
            } else {
                previous->next = current->next;
                if (current->next != NULL)
                    current->next->previous = previous;
            }
            delete current;
            length=length-1;
            cout << "Node with value " << value << " deleted." <<
endl;
            return;
        }
        previous = current;
        current = current->next;
    }
    cout << "Node with value " << value << " not found." << endl;
}
```

d) length() : informs the user of the length of the linked list

```
int length_of_linked_list() {
    return length;
}
```

e) traverse() simply prints the elements of the list either in increasing or decreasing order

```
void traverse(bool increasingOrder) {
    if (length == 0) {
        cout << "List is empty." << endl;
```

```
            return;
        }

        node* current;
        if (increasingOrder) {
            current = start;
            cout << "List elements in increasing order: ";
            while (current != NULL) {
                cout << current->data << " ";
                current = current->next;
            }
        } else {
            current = start;
            while (current->next != NULL)
                current = current->next;
            cout << "List elements in decreasing order: ";
            while (current != NULL) {
                cout << current->data << " ";
                current = current->previous;
            }
        }
        cout << endl;
    }
```

A main( ) function that:

- **a.** Text calls upon an external file containing the numbers [1, 3, 7, 99, 101, 103,107] (you can create a text file yourself.
- **b.** creates and populates the linked list with all the elements presented to it in the text file of part (2a)
- **c.** 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) {
                    create(num);
                } else {
                    add(num);
                }
            }
            inputFile.close();
        } else {
            cout << "Unable to open numbers.txt" << endl;
            return 1;
        }
        // User interaction loop
        int choice;
        do {
            cout << "-------------------------------------------------
-----------------" << endl;
            cout << "Dear User " << endl;
            cout << "Choose an 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 save to file" << endl;
            cout << "Enter your choice: ";cin >> choice;cout<<endl;
            switch (choice) {
                case 1: {
                    int newValue;
                    cout << "Enter the value to add: ";cin >>
newValue;cout<<endl;
                    add(newValue);
                    break;
                }
                case 2: {
                    int newValue;
                    cout << "Enter the value to add: ";cin >>
newValue;cout<<endl;
                    deleteNode(newValue);
```

```
                break;
            }
            case 3: {
                int order;
                cout << "Enumerate the elements of the list:" <<
endl;
                cout << "1. Deccreasing order" << endl;
                cout << "2. Increasing order" << endl;
                cout << "Enter your choice: ";cin >>
order;cout<<endl;
                traverse(order == 1);
                break;
            }
            case 4: {
                cout << "Length of the list: " <<
length_of_linked_list() << endl;
                break;
            }
            case 5: {
                saveListToFile();
                cout << "Linked list saved to output.txt. Exiting..."
<< endl;
                break;
            }
            default:
                cout << "Invalid choice. Please enter a valid
option." << endl;
        }
    } while (choice != 5);


    return 0;
}
```

Output:

Command prompt:

```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\AhmedShafique>D:

D:\>cd "6th Semester"

D:\6th Semester>cd DSA

D:\6th Semester\DSA>cd Lab1

D:\6th Semester\DSA\Lab1>cd DSA_lab

D:\6th Semester\DSA\Lab1\DSA_lab>double_ll.exe double_ll.cpp dli.txt
```

Fig:01 Running on command prompt

```
--------------------------------------------------------------------
Dear User
Choose an action:
[1]. Add a node
[2]. Delete a node
[3]. Enumerate the elements of the list
[4]. Length of the list
[5]. End and save to file
Enter your choice:
```

Fig: 02 Accepting user choice

```
--------------------------------------------------------------------
Dear User
Choose an action:
[1]. Add a node
[2]. Delete a node
[3]. Enumerate the elements of the list
[4]. Length of the list
[5]. End and save to file
Enter your choice: 3

Enumerate the elements of the list:
1. Deccreasing order
2. Increasing order
Enter your choice: 1

List elements in increasing order: 107 103 101 99 9 3 1
```

Fig: 03 Traversing decreasingly

```
--------------------------------------------------------------------
Dear User
Choose an action:
[1]. Add a node
[2]. Delete a node
[3]. Enumerate the elements of the list
[4]. Length of the list
[5]. End and save to file
Enter your choice: 3

Enumerate the elements of the list:
1. Deccreasing order
2. Increasing order
Enter your choice: 2

List elements in decreasing order: 1 3 9 99 101 103 107
```

Fig: 04 Traversing Increasingly

```
--------------------------------------------------------------------
Dear User
Choose an action:
[1]. Add a node
[2]. Delete a node
[3]. Enumerate the elements of the list
[4]. Length of the list
[5]. End and save to file
Enter your choice: 1

Enter the value to add: 45
```

Fig: 05 Adding node

```
--------------------------------------------------------------------
Dear User
Choose an action:
[1]. Add a node
[2]. Delete a node
[3]. Enumerate the elements of the list
[4]. Length of the list
[5]. End and save to file
Enter your choice: 2

Enter the value to add: 45

Node with value 45 deleted.
```

Fig: 06 Deleting node

```
--------------------------------------------------------------------
Dear User
Choose an action:
[1]. Add a node
[2]. Delete a node
[3]. Enumerate the elements of the list
[4]. Length of the list
[5]. End and save to file
Enter your choice: 4

Length of the list: 8
```

Fig: 07 Linked lists length

```
--------------------------------------------------------------------
Dear User
Choose an action:
[1]. Add a node
[2]. Delete a node
[3]. Enumerate the elements of the list
[4]. Length of the list
[5]. End and save to file
Enter your choice: 5

Linked list saved to output.txt. Exiting...
```

Fig: 08 Saving Output text file

Methodology:

First, we added required header files to run the program which are:

```
#include <iostream>
#include <new>
#include <fstream>
using namespcace std;
```

According to the requirement of making structure for double linked list we code the structure of double linked list which is as follows:

```
struct node
{
     int data ;
    node* previous ;
    node * next ;
};
```

Then we initialize the double linked list to zero because there is no sign of double linked list initially using following code:

```
int length = 0;
node* start = NULL;
```

We declare them as global because now these can easily calls in any function so there is no need to additionally passed them into function as a argument. Then we coded the `create( )` function to create the node as user input text file. Then we coded `traverse( )` function, the traverse function prints the value in whole linked list and we set that if there no node then it return nothing and we use while loop and pass the condition that it stops where the next structure pointer points to `NULL`.

We coded the `add( )` function to add additional nodes from the user and then, we coded another function called `delete ( )` to delete the node of double linked list that user want to delete based on the value of the node input by the user.

We coded `size_of_linked_list` function to calculate the length of our double linked list, it same as some portion of traverse function.

At last we coded the last part of program in which user can save the double linked list data into the output text file. So, We initialize the file input in main function then we pass one condition that if there is no sign of linked list and it initialize to zero then it calls the `create` function to create the node in the linked list and initialize it to 1. Else it `add` the nodes in the double linked list using add function.

Also we give choice to user to save it linked lists data that the user ran into output text file using `ofstream`, here is the code below:

```
void saveListToFile() {
    ofstream outputFile("dlo.txt");
    if (outputFile.is_open()) {
        node *temp = start;
        while (temp != nullptr) {
            outputFile << temp->data << " ";
            temp = temp->next;
```

```
        }
        outputFile.close();
    } else {
        cout << "Unable to open output.txt for writing" << endl;
    }
}
```