COMSATS University Islamabad, Vehari Campus

Department of Computer Science

Class: BCS-SP22 **Submission Deadline: 9 Oct 2023**

Subject: Data Structures and Algorithms-Lab **Instructor: Yasmeen Jana**

Max Marks: 20 **Reg. No: SP22-BCS-038(Ahsan)**

Email: yasmeenjana@cuivehari.edu.pk

You can ask queries related to Lab Activities on the above email.

Activity 1:

Create a function to display linked list output as below:

```
#include <iostream>
using namespace std;

class Node {
public:
   int data;
   Node* next;
```

```
    Node(int value) {
        data = value;
        next = nullptr;
    }
};

class LinkedList {
private:
    Node* head;

public:
    LinkedList() {
        head = nullptr;
    }

    // Function to add a node at the end of the list
    void insertAtEnd(int value) {
        Node* newNode = new Node(value);
        if (head == nullptr) {
            head = newNode;
        } else {
            Node* current = head;
            while (current->next != nullptr) {
                current = current->next;
            }
            current->next = newNode;
        }
    }
```

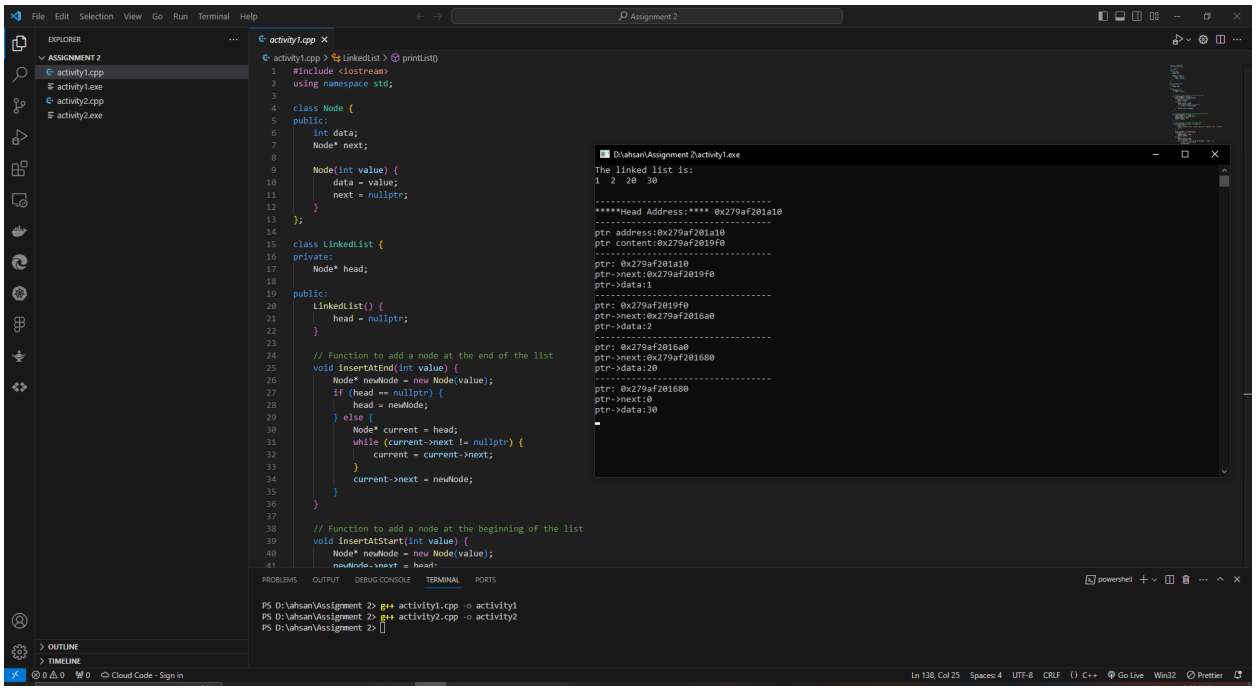
```
// Function to add a node at the beginning of the list
void insertAtStart(int value) {
    Node* newNode = new Node(value);
    newNode->next = head;
    head = newNode;
}

// Function to add a node at a specific index
void insertAtIndex(int value, int index) {
    if (index < 0) {
        cout << "Invalid index. Cannot insert at a negative index." << endl;
        return;
    }

    Node* newNode = new Node(value);
    if (index == 0) {
        newNode->next = head;
        head = newNode;
    } else {
        Node* current = head;
        int currentIndex = 0;
        while (current != nullptr && currentIndex < index - 1) {
            current = current->next;
            currentIndex++;
        }
        if (current == nullptr) {
            cout << "Invalid index. Cannot insert at the specified index." << endl;
            return;
```

```
        }
        newNode->next = current->next;
        current->next = newNode;
    }
}

// Function to delete a node at a specific index
void deleteAtIndex(int index) {
    if (index < 0) {
        cout << "Invalid index. Cannot delete at a negative index." << endl;
        return;
    }

    if (head == nullptr) {
        cout << "List is empty. Cannot delete from an empty list." << endl;
        return;
    }

    if (index == 0) {
        Node* temp = head;
        head = head->next;
        delete temp;
    } else {
        Node* current = head;
        int currentIndex = 0;
        while (current->next != nullptr && currentIndex < index - 1) {
            current = current->next;
            currentIndex++;
        }
```

```
        if (current->next == nullptr) {
            cout << "Invalid index. Cannot delete at the specified index." << endl;
            return;
        }
        Node* temp = current->next;
        current->next = current->next->next;
        delete temp;
    }
}

//Function to del node at first
void deleteAtFirst(){
    Node * temp = head;
    head = (head->next);
    delete temp;
}

//Function to delete at end
void deleteAtEnd(){
    Node * temp=head;
    Node * newlast=temp;
    while (temp->next!= nullptr)
    {
        newlast= temp;
        temp = temp->next;
    }

    newlast->next= nullptr;
    delete temp;
```

```
}

// Function to print the entire linked list as given.
void printList() {
   cout<<"The linked list is:"<<endl;
   Node* current = head;
   while (current != nullptr) {
      cout << current->data << "  ";
      current = current->next;
   }
   cout<<"\n\n----------------------------------"<<endl;
   cout<<"****Head Address:*** " <<head<<endl;

   Node* ptr =head;
   cout<<"----------------------------------"<<endl;
   cout<<"ptr address:"<<ptr<<endl;
   cout<<"ptr content:"<<ptr->next<<endl;
   while(ptr != nullptr){
      cout<<"----------------------------------"<<endl;
      cout<<"ptr: "<<ptr<<endl;
      cout<<"ptr->next:"<<ptr->next<<endl;
      cout<<"ptr->data:"<<ptr->data<<endl;
      ptr= ptr->next;

   }
```

```
    }
};

int main() {
    LinkedList myList;


    myList.insertAtStart(30);
    myList.insertAtStart(20);
    myList.insertAtStart(2);
    myList.insertAtStart(1);

    myList.printList();
    getchar();
    return 0;
}
```



Activity 2:

Write a program that will implement single, doubly, and circular linked link list operations by showing a menu to the user.

The menu should be:

Which linked list you want:

1: Single

2: Double

3: Circular

After the option is chosen by the user:

Which operation you want to perform:

1: Insertion

2: Deletion

3: Display

4: Reverse

4: Seek

5: Exit

Let's suppose, the user has chosen the insertion option then the following menu should be displayed:

1: insertion at beginning

2: insertion at end

3: insertion at the specific data node

A sample output screenshot is below:

Solution:

```
#include <iostream>

using namespace std;
```

```
// Node for linked lists
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
private:
    Node* head;
    Node* tail;
public:
    LinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    // Function for insertion
    void insertAtBeginning(int value) {
        Node* newNode = new Node;
        newNode->data = value;
        newNode->next = head;
        newNode->prev = nullptr;

        if (head != nullptr) {
            head->prev = newNode;
        }

        head = newNode;
```

```
    if (tail == nullptr) {
        tail = newNode;
    }
}

// Function for insertion at end
void insertAtEnd(int value) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->next = nullptr;

    if (tail != nullptr) {
        tail->next = newNode;
        newNode->prev = tail;
    }

    tail = newNode;

    if (head == nullptr) {
        head = newNode;
    }
}

// Function to insert at specific location
void insertAfter(int target, int value) {
    Node* newNode = new Node;
    newNode->data = value;
```

```
    Node* current = head;

    while (current != nullptr && current->data != target) {
        current = current->next;
    }

    if (current != nullptr) {
        newNode->next = current->next;
        current->next = newNode;
        newNode->prev = current;

        if (newNode->next != nullptr) {
            newNode->next->prev = newNode;
        }

        if (current == tail) {
            tail = newNode;
        }
    }
}

// Function for deletion of data
void deleteNode(int value) {
    Node* current = head;
    Node* previous = nullptr;

    while (current != nullptr && current->data != value) {
        previous = current;
        current = current->next;
```

```
    }

    if (current != nullptr) {
        if (previous != nullptr) {
            previous->next = current->next;
        } else {
            head = current->next;
        }

        if (current == tail) {
            tail = previous;
        }

        delete current;
    }
}

// Function to print the list
void display() {
    Node* current = head;

    while (current != nullptr) {
        cout << current->data << " ";
        current = current->next;
    }

    cout << endl;
}
```

```
// Function to reverse the linked list
void reverse() {
    cout<<"\nReversing the list...."<<endl;
    Node* current = head;
    Node* prev = nullptr;
    Node* next = nullptr;

    while (current != nullptr) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }

    head = prev;
}

// Function to seek a specific data value
bool seek(int value) {
    Node* current = head;

    while (current != nullptr) {
        if (current->data == value) {
            return true;
        }
        current = current->next;
    }

    return false;
```

```
    }
};

int main() {
    int insertValue,choice, subChoice;
    LinkedList sList, dList, cList;



    do { //do fpr list selection operation
        cout << "Which linked list you want:\n"
             << "1: Single\n"
             << "2: Double\n"
             << "3: Circular\n"
             << "Select: ";
        cin >> choice;

        switch (choice) {
            case 1:
                do { //after selection of singly
                    cout << "Which operation you want to perform:\n"
                         << "1: Insertion\n"
                         << "2: Deletion\n"
                         << "3: Display\n"
                         << "4: Reverse\n"
                         << "5: Seek\n"
                         << "6: Exit\n"
                         << "Select: ";
                    cin >> subChoice;
```

```
switch (subChoice) {
    case 1:
        int typeofinsertion;
        cout<<"1. Insertion at beigning\n"
            <<"2. Insertion at end\n"
            <<"3. Insertion at specific data value\n"
            <<"Chose your option: ";
        cin>>typeofinsertion;
        switch(typeofinsertion){
            case 1:

                cout << "Enter the value to insert: ";
                cin >> insertValue;
                sList.insertAtBeginning(insertValue);
                cout<<"\n Items in list: ";
                sList.display();
                break;
            case 2:
                cout << "Enter the value to insert: ";
                cin >> insertValue;
                sList.insertAtEnd(insertValue);
                cout<<"\n Items in list: ";
                sList.display();
                break;
            case 3:
                int targetval;
                cout << "Enter the value to insert: ";
                cin >> insertValue;
                cout<<"ENter the value where you want to insert: ";
```

```
            cin>>targetval;
            sList.insertAfter(targetval,insertValue);
            cout<<"\n Items in list: ";
            sList.display();
            break;
        default:
            cout<<"You didn't selected appropiate option"<<endl;
            break;
    }
    break;
case 2:
    int deleteValue;
    cout << "Enter the value to delete: ";
    cin >> deleteValue;
    sList.deleteNode(deleteValue);
    cout<<"Value deleted"<<endl;
    break;
case 3:
    cout << "Single Linked List: ";
    sList.display();
    break;
case 4:
    sList.reverse();
    break;
case 5:
    int seekValue;
    cout << "Enter the value to seek: ";
    cin >> seekValue;
    if (sList.seek(seekValue)) {
```

```
                cout << "Value found in Single Linked List.\n";
            } else {
                cout << "Value not found in Single Linked List.\n";
            }
            break;
        case 6:
            cout << "Exiting Single Linked List menu.\n";
            break;
        default:
            cout << "Invalid choice. Please try again.\n";
            break;
    }
  } while (subChoice != 6);
  break;

case 2:
  do { //after slection of doubly
    cout << "Which operation you want to perform:\n"
        << "1: Insertion\n"
        << "2: Deletion\n"
        << "3: Display\n"
        << "4: Reverse\n"
        << "5: Seek\n"
        << "6: Exit\n"
        << "Enter your choice : ";
    cin >> subChoice;

    switch (subChoice) {
        case 1:
```

```
int typeofinsertion;
cout<<"1. Insertion at beigning\n"
   <<"2. Insertion at end\n"
   <<"3. Insertion at specific data value\n"
   <<"Chose your option: ";
cin>>typeofinsertion;
switch(typeofinsertion){
   case 1:
      cout << "Enter the value to insert: ";
      cin >> insertValue;
      dList.insertAtBeginning(insertValue);
      cout<<"\n Items in list: ";
      dList.display();
      break;
   case 2:
      cout << "Enter the value to insert: ";
      cin >> insertValue;
      dList.insertAtEnd(insertValue);
      cout<<"\n Items in list: ";
      dList.display();
      break;
   case 3:
      int targetval;
      cout << "Enter the value to insert: ";
      cin >> insertValue;
      cout<<"ENter the value where you want to insert: ";
      cin>>targetval;
      dList.insertAfter(targetval,insertValue);
```

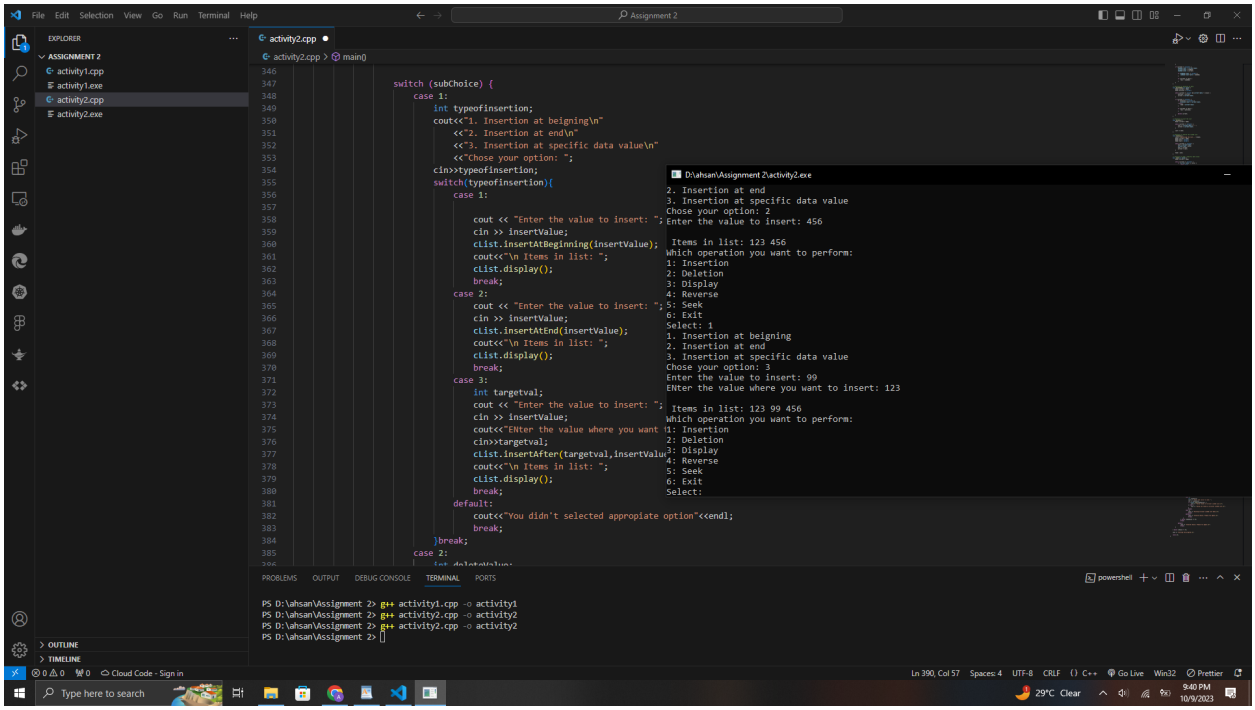
```
            cout<<"\n Items in list: ";
            dList.display();
            break;
        default:
            cout<<"You didn't selected appropiate option"<<endl;
            break;
    }break;
case 2:
    int deleteValue;
    cout << "Enter the value to delete: ";
    cin >> deleteValue;
    dList.deleteNode(deleteValue);
    cout<<"Value Deleted"<<endl;
    break;
case 3:
    cout << "Double Linked List: ";
    dList.display();
    break;
case 4:
    dList.reverse();
    break;
case 5:
    int seekValue;
    cout << "Enter the value to seek: ";
    cin >> seekValue;
    if (dList.seek(seekValue)) {
        cout << "Value found in Double Linked List.\n";
    } else {
        cout << "Value not found in Double Linked List.\n";
```

```
            }
            break;
        case 6:
            cout << "Exiting Double Linked List menu.\n";
            break;
        default:
            cout << "Invalid choice. Please try again.\n";
            break;
      }
   } while (subChoice != 6);
   break;

case 3:
   do { //after selection of circular
      cout << "Which operation you want to perform:\n"
         << "1: Insertion\n"
         << "2: Deletion\n"
         << "3: Display\n"
         << "4: Reverse\n"
         << "5: Seek\n"
         << "6: Exit\n"
         << "Enter your choice: ";
      cin >> subChoice;
      switch (subChoice) {
         case 1:
            int typeofinsertion;
            cout<<"1. Insertion at beigning\n"
               <<"2. Insertion at end\n"
               <<"3. Insertion at specific data value\n"
```

```
   <<"Chose your option: ";
cin>>typeofinsertion;
switch(typeofinsertion){
   case 1:

      cout << "Enter the value to insert: ";
      cin >> insertValue;
      cList.insertAtBeginning(insertValue);
      cout<<"\n Items in list: ";
      cList.display();
      break;
   case 2:
      cout << "Enter the value to insert: ";
      cin >> insertValue;
      cList.insertAtEnd(insertValue);
      cout<<"\n Items in list: ";
      cList.display();
      break;
   case 3:
      int targetval;
      cout << "Enter the value to insert: ";
      cin >> insertValue;
      cout<<"ENter the value where you want to insert: ";
      cin>>targetval;
      cList.insertAfter(targetval,insertValue);
      cout<<"\n Items in list: ";
      cList.display();
      break;
   default:
```

```
            cout<<"You didn't selected appropiate option"<<endl;
            break;
    }break;
case 2:
    int deleteValue;
    cout << "Enter the value to delete: ";
    cin >> deleteValue;
    cList.deleteNode(deleteValue);
    cout<<"Value deleted"<<endl;
    break;
case 3:
    cout << "Circular Linked List: ";
    cList.display();
    break;
case 4:
    cout << "Cannot reverse a circular linked list.\n";
    break;
case 5:
    int seekValue;
    cout << "Enter the value to seek: ";
    cin >> seekValue;
    if (cList.seek(seekValue)) {
        cout << "Value found in Circular Linked List.\n";
    } else {
        cout << "Value not found in Circular Linked List.\n";
    }
    break;
case 6:
    cout << "Exiting Circular Linked List menu.\n";
```

```
                    break;
                default:
                    cout << "Invalid choice. Please try again.\n";
                    break;
            }
        } while (subChoice != 6);
        break;
    default:
        cout << "Invalid choice. Please try again.\n";
        break;
    }
  } while (choice != 4);
  cout << "Exiting the program.\n";
  return 0;
}
```


File Edit Selection View Go Run Terminal Help
Assignment 2
EXPLORER
ASSIGNMENT 2
activity1.cpp
activity1.exe
activity2.cpp
activity2.exe
activity2.cpp
activity2.cpp > main()
switch (subChoice) {
case 1:
int typeofinsertion;
cout<<"1. Insertion at beigning\n"
<<"2. Insertion at end\n"
<<"3. Insertion at specific data value\n"
<<"Chose your option: ";
cin>>typeofinsertion;
switch(typeofinsertion){
case 1:
cout << "Enter the value to insert: ";
cin >> insertValue;
cList.insertAtBeginning(insertValue);
cout<<"\n Items in list: ";
cList.display();
break;
case 2:
cout << "Enter the value to insert: ";
cin >> insertValue;
cList.insertAtEnd(insertValue);
cout<<"\n Items in list: ";
cList.display();
break;
case 3:
int targetval;
cout << "Enter the value to insert: ";
cin >> insertValue;
cout<<"ENter the value where you want to insert: ";
cin>>targetval;
cList.insertAfter(targetval,insertValue);
cout<<"\n Items in list: ";
cList.display();
break;
default:
cout<<"You didn't selected appropiate option"<<endl;
break;
}break;
case 2:
D:\ahsan\Assignment 2\activity2.exe
2. Insertion at end
3. Insertion at specific data value
Chose your option: 2
Enter the value to insert: 456
Items in list: 123 456
Which operation you want to perform:
1: Insertion
2: Deletion
3: Display
4: Reverse
5: Seek
6: Exit
Select: 1
1. Insertion at beigning
2. Insertion at end
3. Insertion at specific data value
Chose your option: 3
Enter the value to insert: 99
ENter the value where you want to insert: 123
Items in list: 123 99 456
Which operation you want to perform:
1: Insertion
2: Deletion
3: Display
4: Reverse
5: Seek
6: Exit
Select:
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
PS D:\ahsan\Assignment 2> g++ activity1.cpp -o activity1
PS D:\ahsan\Assignment 2> g++ activity2.cpp -o activity2
PS D:\ahsan\Assignment 2> g++ activity2.cpp -o activity2
PS D:\ahsan\Assignment 2>
OUTLINE
TIMELINE
Cloud Code - Sign in
Ln 390, Col 57 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier
Type here to search
29°C Clear
9:40 PM
10/9/2023