

C++

Assignment – 9

Name – Kishan R Vaghamashi

Student ID – 202312014

1)

```
#define _USE_MATH_DEFINES
#include <cmath>
#include <iostream>
using namespace std;
template <typename T>
struct LinkedList
{
    struct Node
    {
        T data;
        unique_ptr<Node> next = nullptr;
        Node(T data)
        {
            this->data = data;
        }
        static auto get_new_node(T data)
        {
            return make_unique<Node>(data);
        }
    };
    unique_ptr<Node> head = nullptr;
    LinkedList() {}
    LinkedList(T data)
    {
        insert(data);
    }
    void print()
    {
        if (head == nullptr)
            return;
        auto traversal = head.get();
        while (true)
        {
            cout << traversal->data << " -> ";
            if (traversal->next != nullptr)
                traversal = traversal->next.get();
            else
                break;
        }
        cout << " NULL " << endl;
    }
    void push(T data)
    {
        auto new_node = Node::get_new_node(data);
```

```

        if (head == nullptr)
        {
            head = move(new_node);
            return;
        }
        auto traversal = head.get();
        while (traversal->next != nullptr)
        {
            traversal = traversal->next.get();
        }
        traversal->next = move(new_node);
    }
    void pop()
    {
        if (head == nullptr)
            return;
        if (head->next == nullptr)
        {
            head.reset();
            return;
        }
        auto traversal = head.get();
        while (traversal->next->next != nullptr)
        {
            traversal = traversal->next.get();
        }
        traversal->next.reset();
    }
};

struct Circle
{
    double rad = 0;
    double area = 0;
    Circle() {}
    Circle(double rad)
    {
        this->rad = rad;
        this->area = calc_area();
    }

private:
    double calc_area()
    {
        auto rad = this->rad;
        return M_PI * rad * rad;
    }
};

```

```

    }
};
ostream &operator<<(ostream &os, Circle c)
{
    os << "(Radius: " << c.rad << " Area: " << c.area << ")";
    return os;
}
int main()
{
    LinkedList<Circle> ll;
    if (ll.head == nullptr)
    {
        cout << "Test 1: Passed (Initially Empty)" << endl;
    }
    else
    {
        cout << "Test 1: Failed (Initially Not Empty)" << endl;
    }
    ll.push(10);
    ll.push(20);
    if (ll.head != nullptr && ll.head->data.rad == 10 &&
        ll.head->next->data.rad == 20)
    {
        cout << "Test 2: Passed (Added Elements)" << endl;
    }
    else
    {
        cout << "Test 2: Failed (Adding Elements)" << endl;
    }
    ll.pop();
    if (ll.head != nullptr && ll.head->data.rad == 10 && ll.head->next ==
    nullptr)
    {
        cout << "Test 3: Passed (Removed Element)" << endl;
    }
    else
    {
        cout << "Test 3: Failed (Removing Element)" << endl;
    }
    ll.push(30);
    ll.push(40);
    cout << "Test 4: ";
    ll.print();
    LinkedList<Circle> ll2;
    ll2.push(2.5);

```

```

    ll2.push(3.5);
    cout << "Test 5: ";
    ll2.print();
    return 0;
}

```

Output:

The screenshot shows the Visual Studio Code editor with a C++ file named `31-doubly_linked_list.cpp`. The code implements a doubly linked list with a `Node` struct and a `LinkedList` class. The `Node` struct contains a `T data` member and a `unique_ptr<Node> next` member. The `LinkedList` class has methods for `insert`, `print`, and `push`. The `print` method traverses the list and prints the data of each node. The `push` method adds a new node to the end of the list. The output window shows the results of the tests:

```

PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Use
rs\kishan.HP-PROBOOK\Desktop\DSA\" ; if ($?) { g++ lc
.cpp -o lc } ; if ($?) { .\lc }
Test 1: Passed (Initially Empty)
Test 2: Passed (Added Elements)
Test 3: Passed (Removed Element)
Test 4: (Radius: 10 Area: 314.159) -> (Radius: 30 Are
a: 2827.43) -> (Radius: 40 Area: 5026.55) -> NULL
Test 5: (Radius: 2.5 Area: 19.635) -> (Radius: 3.5 Ar
ea: 38.4845) -> NULL
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA>

```

2)

```

#include <sstream>
#include <fstream>
#include <iostream>
#include <string>
#include <bits/stdc++.h>
using namespace std;
template <typename T>
struct LinkedList
{
    struct Node
    {
        T data;
        unique_ptr<Node> next = nullptr;
        Node(T data)
        {
            this->data = data;
        }
    };
};

```

```

        static auto get_new_node(T data)
        {
            return make_unique<Node>(data);
        }
};
unique_ptr<Node> head = nullptr;
LinkedList() {}
LinkedList(T data)
{
    insert(data);
}
void print()
{
    if (head == nullptr)
        return;
    auto traversal = head.get();
    while (true)
    {
        cout << traversal->data << " -> ";
        if (traversal->next != nullptr)
            traversal = traversal->next.get();
        else
            break;
    }
    cout << " NULL " << endl;
}
void push(T data)
{
    auto new_node = Node::get_new_node(data);
    if (head == nullptr)
    {
        head = move(new_node);
        return;
    }
    auto traversal = head.get();
    while (traversal->next != nullptr)
    {
        traversal = traversal->next.get();
    }
    traversal->next = move(new_node);
}
void pop()
{
    if (head == nullptr)
        return;

```

```

        if (head->next == nullptr)
        {
            head.reset();
            return;
        }
        auto traversal = head.get();
        while (traversal->next->next != nullptr)
        {
            traversal = traversal->next.get();
        }
        traversal->next.reset();
    }
};

struct NobelPrize
{
    string name;
    int year;
    string category;
};

ostream &operator<<(ostream &os, NobelPrize n)
{
    os << "(Name: " << n.name << " Year: " << n.year << " Category: " <<
n.category << ")" << endl;
    return os;
}

int main()
{
    LinkedList<NobelPrize> nobelPrizeList;
    ifstream inputFile("nobel_prize_data.txt");
    if (!inputFile.is_open())
    {
        cerr << "Failed to open the file." << endl;
        return 1;
    }
    string line;
    int lineNum = 0;
    while (getline(inputFile, line))
    {
        stringstream ss(line);
        string name;
        int year;
        string category;
        char delim = '|';
        getline(ss, name, delim);
        ss >> year;
    }
}

```

```

getline(ss, category, delim);
lineNum++;
if (lineNum > 2)
{
    NobelPrize prize = {name, year, category};
    nobelPrizeList.push(prize);
}
}
nobelPrizeList.print();
return 0;
}

```

Output:

```

C:\Users\kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Users\kishan.HP-PROBOOK\Desktop\DSA" & if ($?) { g++ lc.cpp -o lc } & if ($?) { .\lc }
(Name: Alfred Nobel Year: 1901 Category: )
-> (Name: Wilhelm Röntgen Year: 1901 Category: )
-> (Name: Jacobus Henricus van't Hoff Year: 1901 Category: )
-> (Name: Emil Adolf von Behring Year: 1901 Category: )
-> (Name: Sully Prudhomme Year: 1901 Category: )
-> (Name: Theodore Roosevelt Year: 1906 Category: )
-> (Name: Henri Becquerel Year: 1903 Category: )
-> (Name: Pierre Curie Year: 1903 Category: )
-> (Name: Marie Curie Year: 1903 Category: )
-> (Name: Svante Arrhenius Year: 1903 Category: )
-> (Name: Niels Ryberg Finsen Year: 1903 Category: )
-> (Name: Frédéric Mistral Year: 1904 Category: )
-> (Name: José Echegaray y Eizaguirre Year: 1904 Category: )
-> (Name: Baron de Coubertin Year: 1904 Category: )
-> (Name: Ivan Pavlov Year: 1904 Category: )
-> (Name: William Ramsay Year: 1904 Category: )
-> (Name: Philipp Lenard Year: 1905 Category: )
-> (Name: Henryk Sienkiewicz Year: 1905 Category: )
-> (Name: Bertha von Suttner Year: 1905 Category: )
-> (Name: Robert Koch Year: 1905 Category: )
-> (Name: Adolf von Baeyer Year: 1905 Category: )
-> (Name: Giosuè Carducci Year: 1906 Category: )
-> (Name: Elhu Root Year: 1906 Category: )
-> (Name: Albert Michelson Year: 1907 Category: )
-> (Name: Eduard Buchner Year: 1907 Category: )
-> (Name: Giosuè Carducci Year: 1907 Category: )
-> (Name: Ernesto Moneta Year: 1907 Category: )
-> (Name: Paul Ehrlich Year: 1908 Category: )
-> (Name: Gabriel Lippmann Year: 1908 Category: )
-> (Name: Ernest Rutherford Year: 1908 Category: )
-> (Name: Rudolf Eucken Year: 1908 Category: )
-> (Name: Klas Pontus Arnoldson Year: 1908 Category: )
-> (Name: Camillo Golgi Year: 1909 Category: )
-> (Name: Albert Einstein Year: 1921 Category: )
-> (Name: Frédéric Joliot-Curie Year: 1935 Category: )
-> (Name: Irène Joliot-Curie Year: 1935 Category: )
-> (Name: James B. Conant Year: 1947 Category: )
-> (Name: William Shockley Year: 1956 Category: )
-> (Name: John Bardeen Year: 1956 Category: )
-> (Name: Walter H. Brattain Year: 1956 Category: )
-> (Name: C.N. Yang Year: 1957 Category: )
-> (Name: Tsung-Dao Lee Year: 1957 Category: )
-> (Name: Owen Chamberlain Year: 1959 Category: )
-> (Name: Emilio G. Segrè Year: 1959 Category: )
-> (Name: Max Perutz Year: 1962 Category: )
-> (Name: John Kendrew Year: 1962 Category: )

```

3)

```

#include <sstream>
#include <fstream>
#include <bits/stdc++.h>
#include <string>
#include <functional>
using namespace std;
template <typename T>
struct LinkedList
{
    struct Node
    {

```



```

    T data;
    unique_ptr<Node> next = nullptr;
    Node(T data)
    {
        this->data = data;
    }
    static auto get_new_node(T data)
    {
        return make_unique<Node>(data);
    }
};
unique_ptr<Node> head = nullptr;
LinkedList() {}
LinkedList(T data)
{
    insert(data);
}
void print()
{
    if (head == nullptr)
        return;
    auto traversal = head.get();
    while (true)
    {
        cout << traversal->data << " -> ";
        if (traversal->next != nullptr)
            traversal = traversal->next.get();
        else
            break;
    }
    cout << " NULL " << endl;
}
void push(T data)
{
    auto new_node = Node::get_new_node(data);
    if (head == nullptr)
    {
        head = move(new_node);
        return;
    }
    auto traversal = head.get();
    while (traversal->next != nullptr)
    {
        traversal = traversal->next.get();
    }
}

```

```

        traversal->next = move(new_node);
    }
    void pop()
    {
        if (head == nullptr)
            return;
        if (head->next == nullptr)
        {
            head.reset();
            return;
        }
        auto traversal = head.get();
        while (traversal->next->next != nullptr)
        {
            traversal = traversal->next.get();
        }
        traversal->next.reset();
    }
};

struct NobelPrize
{
    string name;
    int year;
    string category;
};

bool CompareNobelPrizesByYear(const NobelPrize &a, const NobelPrize &b)
{
    return a.year >= b.year;
}

ostream &operator<<(ostream &os, NobelPrize n)
{
    os << "(Name: " << n.name << " Year: " << n.year << " Category: " <<
n.category << ")" << endl;
    return os;
}

void insertIntoSorted(LinkedList<NobelPrize> &list, const NobelPrize &prize,
                     std::function<bool(const NobelPrize &, const NobelPrize
&)> compareFunction)
{
    auto new_node = LinkedList<NobelPrize>::Node::get_new_node(prize);
    if (list.head == nullptr || compareFunction(prize, list.head->data))
    {
        new_node->next = move(list.head);
        list.head = move(new_node);
    }
}

```

```

else
{
    auto prevSorted = list.head.get();
    auto current = list.head.get();
    while (current != nullptr && compareFunction(current->data,
                                                prize))
    {
        prevSorted = current;
        current = current->next.get();
    }
    new_node->next = move(prevSorted->next);
    prevSorted->next = move(new_node);
}
}

int main()
{
    LinkedList<NobelPrize> nobelPrizeList;
    LinkedList<NobelPrize> sortedNobelPrizeList;
    auto SortNobelPrizes = [&](std::function<bool(const NobelPrize &, const
NobelPrize &)> compareFunction)
    {
        sortedNobelPrizeList.head.reset();
        auto current = nobelPrizeList.head.get();
        while (current)
        {
            auto next = current->next.get();
            insertIntoSorted(sortedNobelPrizeList, current->data,
                            compareFunction);
            current = next;
        }
    };

    ifstream inputFile("nobel_prize_data.txt");
    if (!inputFile.is_open())
    {
        cerr << "Failed to open the file." << endl;
        return 1;
    }

    string line;
    int lineNum = 0;
    while (getline(inputFile, line))
    {
        stringstream ss(line);
        string name;
        int year;
        string category;

```

```

char delim = '|';
getline(ss, name, delim);
ss >> year;
getline(ss, category, delim);
lineNum++;
if (lineNum > 2)
{
    NobelPrize prize = {name, year, category};
    nobelPrizeList.push(prize);
}
}
SortNobelPrizes(CompareNobelPrizesByYear);
sortedNobelPrizeList.print();
return 0;
}

```

Output:

```

C++ lccpp M X
You, 54 seconds ago | 1 author (You)
1 #include <sstream>
2 #include <fstream>
3 #include <bits/stdc++.h>
4 #include <string>
5 #include <functional>
6 using namespace std;
7 template <typename T>
8 struct LinkedList
9 {
10     struct Node
11     {
12         T data;
13         unique_ptr<Node> next = nullptr;
14         Node(T data)
15         {
16             this->data = data;
17         }
18         static auto get_new_node(T data)
19         {
20             return make_unique<Node>(data);
21         }
22     };
23     unique_ptr<Node> head = nullptr;
24     LinkedList() {}
25     LinkedList(T data)
26     {
27         insert(data);
28     }
29     void print()
30     {
31         if (head == nullptr)
32             return;
33         auto traversal = head.get();
34         while (true)
35         {
36             cout << traversal->data << " -> ";
37             if (traversal->next != nullptr)

```

```

PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Users\kishan.HP-PROBOOK\Desktop\DSA" ; if ($?) { g++ lc.cpp -o lc ; }
if ($?) { .\lc }
(Name: Malala Yousafzai Year: 2014 Category: )
-> (Name: Barack Obama Year: 2009 Category: )
-> (Name: F.W. de Klerk Year: 1993 Category: )
-> (Name: Nelson Mandela Year: 1993 Category: )
-> (Name: Aung San Suu Kyi Year: 1991 Category: )
-> (Name: Martin Luther King Jr. Year: 1964 Category: )
-> (Name: John Hendrew Year: 1962 Category: )
-> (Name: Max Perutz Year: 1962 Category: )
-> (Name: Emilio G. Segrè Year: 1959 Category: )
-> (Name: Owen Chamberlain Year: 1959 Category: )
-> (Name: Tsung-Dao Lee Year: 1957 Category: )
-> (Name: C.N. Yang Year: 1957 Category: )
-> (Name: Walter H. Brattain Year: 1956 Category: )
-> (Name: John Bardeen Year: 1956 Category: )
-> (Name: William Shockley Year: 1956 Category: )
-> (Name: James B. Conant Year: 1947 Category: )
-> (Name: Irène Joliot-Curie Year: 1935 Category: )
-> (Name: Frédéric Joliot-Curie Year: 1935 Category: )
-> (Name: Albert Einstein Year: 1921 Category: )
-> (Name: Camillo Golgi Year: 1909 Category: )
-> (Name: Klas Pontus Arnoldson Year: 1908 Category: )
-> (Name: Rudolf Eucken Year: 1908 Category: )
-> (Name: Ernest Rutherford Year: 1908 Category: )
-> (Name: Gabriel Lippmann Year: 1908 Category: )
-> (Name: Paul Ehrlich Year: 1908 Category: )
-> (Name: Ernesto Moneta Year: 1907 Category: )
-> (Name: Giosuè Carducci Year: 1907 Category: )
-> (Name: Eduard Buchner Year: 1907 Category: )
-> (Name: Albert Nicholson Year: 1907 Category: )
-> (Name: Elihu Root Year: 1906 Category: )
-> (Name: Giosuè Carducci Year: 1906 Category: )
-> (Name: Theodore Roosevelt Year: 1906 Category: )
-> (Name: Philipp Lenard Year: 1905 Category: )
-> (Name: Henryk Sienkiewicz Year: 1905 Category: )
-> (Name: Bertha von Suttner Year: 1905 Category: )
-> (Name: Robert Koch Year: 1905 Category: )
-> (Name: Adolf von Baeyer Year: 1905 Category: )
-> (Name: Frédéric Mistral Year: 1904 Category: )
-> (Name: José Echegaray y Eizaguirre Year: 1904 Category: )
-> (Name: Baron de Coubertin Year: 1904 Category: )
-> (Name: Ivan Pavlov Year: 1904 Category: )
-> (Name: William Ramsay Year: 1904 Category: )
-> (Name: Henri Becquerel Year: 1903 Category: )
-> (Name: Pierre Curie Year: 1903 Category: )
-> (Name: Marie Curie Year: 1903 Category: )

```