

Lab Practice Week 4

Name : Heng SovannReach

Exercise 1 :

```
1 #include<iostream>
2 #include<string>
3
4 class Stack {
5 private:
6     struct Node {
7         std::string data;
8         Node* next;
9         Node* prev;
10        Node(std::string value) : data(value), next(nullptr), prev(nullptr) {}
11    };
12    Node* top;
13 public:
14    Stack() {
15        top = nullptr;
16    }
17 ~Stack() {
18     Node* current = top;
19     Node* nextNode;
20     while (current != nullptr) {
21         nextNode = current->next;
22         delete current;
23         current = nextNode;
24     }
25     top = nullptr;
26 }
27 void push(std::string value) {
28     Node* newNode = new Node(value);
29
30     if (top != nullptr) {
31         top->prev = newNode;
32     }
33
34     newNode->next = top;
35     top = newNode;
36 }
37
38 void pop() {
39     if (top == nullptr) {
40         std::cout << "Stack underflow\n";
41     }
42 }
```

```

42     }
43     Node* temp = top;
44     top = top->next;
45     if (top != nullptr) {
46         top->prev = nullptr;
47     }
48     delete temp;
49 }
50 bool isEmpty() {
51     return top == nullptr;
52 }
53 void displayBackward() {
54     if (isEmpty()) {
55         std::cout << "Stack is empty\n";
56         return;
57     }
58     std::cout << "Stack : ";
59     Node* current = top;
60     while (current != nullptr) {
61         std::cout << current->data << " ";
62         current = current->next;
63     }
64     std::cout << "\n";
65 }
66 };
67 int main() {
68     Stack s;
69     std::string word = "HELLO";
70     for (int i = 0 ; i < 5; i++) {
71         s.push(std::string(1, word[i]));
72     }
73     s.displayBackward();
74     s.pop();
75     s.pop();
76     s.displayBackward();
77
78     return 0;

```

Output :

```

Microsoft-MIEngine-Out-phnupjhb.vwm' '--s'
Stack : O L L E H
Stack : L E H
PS C:\Users\USER\Desktop\c++> █

```

Exercise2 :

```
1 #include <iostream>
2 class Node{
3 public:
4     int id;
5     Node* next;
6     Node(int i):id(i),next(nullptr){}
7 };
8 class Queue{
9 private:
10    Node* front;
11    Node* rear;
12 public:
13     // Constructor for front and rear
14     Queue(){
15         front = nullptr;
16         rear = nullptr;
17     }
18     ~Queue(){
19         Node* temp = front;
20         if(!front){
21             std::cout<<"Empty list \n";
22             return;
23         }
24         while (temp != nullptr){
25             Node* next_node = temp->next;
26             delete temp;
27             temp = next_node;
28         }
29         front = nullptr;
30         std::cout<<"Everything is cleared yeppiiiiii";
31     }
32     void display(){
33         if(!front){
34             std::cout<<"Empty list \n";
35             return;
36         }
37         Node* temp = front;
38         while (temp != nullptr){
39             std::cout<<temp->id<<" ";
40             temp = temp->next;
```

```

44     void enqueue(int i){
45         Node* new_Node = new Node(i);
46         if(!front){
47             front = rear = new_Node;
48             return;
49         }
50         // Use rear to point to new node
51         rear->next = new_Node;
52         // Make rear point to new node that is just created
53         rear = new_Node;
54     }
55     void dequeue(){
56         if(!front){
57             std::cout<<"List is empty \n";
58             return;
59         }
60         Node* temp = front->next;
61         Node* current = front;
62         delete current;
63         front = temp;
64     }
65 };
66 int main(){
67     Queue q ;
68     for (int i = 0 ; i < 5; i ++){
69         q.enqueue(i*i);
70     }
71     q.display();
72     q.dequeue();
73     q.display();
74     q.dequeue();
75     q.display();
76     return 0;
77 }
78
79

```

Output :

OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS	PROBLEMS
PS C:\Users\USER\Desktop\c++> & 'c:\Users\USER\.vscode\extensions\ms-vscode				
Microsoft-MIEngine-Out-n45oj20j.pzp' '--stderr=Microsoft-MIEngine-Error-m:				
0 1 4 9 16				
1 4 9 16				
4 9 16				
Everything is cleared yeppiiiiii				
PS C:\Users\USER\Desktop\c++>				

Exercise 3 :

```
1 #include <iostream>
2 #include <string>
3 using namespace std ;
4 struct Node {
5     std::string data;
6     Node* next;
7 };
8
9 class Stack {
10 private:
11     Node* top;
12 public:
13     Stack() : top(nullptr) {}
14
15     ~Stack() {
16         while (top != nullptr) {
17             pop();
18         }
19     }
20
21     void push(const string& item) {
22         Node* newNode = new Node{item, top};
23         top = newNode;
24     }
25
26     string pop() {
27         if (isEmpty()) {
28             return "";
29         }
30         std::string data = top->data;
31         Node* temp = top;
32         top = top->next;
33         delete temp;
34         return data;
35     }
36
37     bool isEmpty() const {
38         return top == nullptr;
39     }
40 }
```

```
42     string peek() const {
43         if (isEmpty()) {
44             return "";
45         }
46         return top->data;
47     }
48 };
49
50 class BrowserHistory {
51 private:
52     Stack backStack;
53     Stack forwardStack;
54     std::string currentUrl;
55
56 public:
57     BrowserHistory() : currentUrl("home") {}
58
59     void visit(const string& url) {
60         if (currentUrl != "home") {
61             backStack.push(currentUrl);
62         }
63         currentUrl = url;
64         while (!forwardStack.isEmpty()) {
65             forwardStack.pop();
66         }
67     }
68
69     string back() {
70         if (backStack.isEmpty()) {
71             return "Cannot go back. Current: " + currentUrl;
72         }
73         forwardStack.push(currentUrl);
74         currentUrl = backStack.pop();
75         return "Current URL: " + currentUrl;
76     }
77 }
```

```

77     string forward() {
78         if (forwardStack.isEmpty()) {
79             return "Cannot go forward. Current: " + currentUrl;
80         }
81         backStack.push(currentUrl);
82         currentUrl = forwardStack.pop();
83         return "Current URL: " + currentUrl;
84     }
85 }
86
87 std::string getCurrentUrl() const {
88     return "Current URL: " + currentUrl;
89 }
90 };
91
92 int main() {
93     BrowserHistory browser;
94
95     browser.visit("www.google.com");
96     browser.visit("www.youtube.com");
97     browser.visit("www.wikipedia.org");
98
99     cout << browser.getCurrentUrl() << endl;
100
101    cout << browser.back() << endl;
102    cout << browser.back() << endl;
103
104    cout << browser.forward() << endl;
105
106    browser.visit("www.gemini.com");
107    cout << browser.getCurrentUrl() << endl;
108
109    cout << browser.forward() << endl;
110
111    cout << browser.back() << endl;
112
113    return 0;

```

Output :

```

PS C:\Users\USER\Desktop\c++> & 'c:\Users\USER\.vscode\Microsoft-MIEngine-Out-tqns3w3b.rad' '--stderr=Microsoft
Current URL: www.wikipedia.org
Current URL: www.youtube.com
Current URL: www.google.com
Current URL: www.youtube.com
Current URL: www.gemini.com
Cannot go forward. Current: www.gemini.com
Current URL: www.youtube.com
PS C:\Users\USER\Desktop\c++>

```

Exercise4 :

```
Untracked
1 #include <iostream>
2
3
4 using namespace std;
5
6 struct Node {
7     string data;
8     Node* next;
9 };
10
11 class Queue {
12 private:
13     Node* front;
14     Node* rear;
15
16 public:
17     Queue() : front(nullptr), rear(nullptr) {}
18
19 ~Queue() {
20     while (front != nullptr) {
21         Node* temp = front;
22         front = front->next;
23         delete temp;
24     }
25     rear = nullptr;
26 }
27
28 void enqueue(const string& item) {
29     Node* newNode = new Node{item, nullptr};
30
31     if (isEmpty()) {
32         front = newNode;
33         rear = newNode;
34     } else {
35         rear->next = newNode;
36         rear = newNode;
37     }
38 }
```

```
38
39     string dequeue() {
40         if (isEmpty()) {
41             return "";
42         }
43         string data = front->data;
44         Node* temp = front;
45         front = front->next;
46         if (front == nullptr) {
47             rear = nullptr;
48         }
49         delete temp;
50         return data;
51     }
52
53     bool isEmpty() const {
54         return front == nullptr;
55     }
56
57     string peek() const {
58         if (isEmpty()) {
59             return "Queue is empty.";
60         }
61         return front->data;
62     }
63 };
64
```

```
65 int main() {
66     Queue bankQueue;
67
68     bankQueue.enqueue("Customer A (Deposit)");
69     cout << "Added: Customer A" << endl;
70     cout << "Next to be served: " << bankQueue.peek() << endl;
71
72     bankQueue.enqueue("Customer B (Withdrawal)");
73     cout << "Added: Customer B" << endl;
74
75     bankQueue.enqueue("Customer C (Loan Inquiry)");
76     cout << "Added: Customer C" << endl;
77     cout << "Next to be served: " << bankQueue.peek() << endl;
78
79     cout << "\n--- Serving Customers ---" << endl;
80     cout << "Served: " << bankQueue.dequeue() << endl;
81     cout << "Served: " << bankQueue.dequeue() << endl;
82     cout << "Next to be served: " << bankQueue.peek() << endl;
83
84     bankQueue.enqueue("Customer D (New Account)");
85     cout << "Added: Customer D" << endl;
86     cout << "Served: " << bankQueue.dequeue() << endl;
87
88     cout << "Is the queue empty? " << (bankQueue.isEmpty() ? "Yes" : "No") << endl;
89
90     cout << "Served: " << bankQueue.dequeue() << endl;
91     cout << "Is the queue empty? " << (bankQueue.isEmpty() ? "Yes" : "No") << endl;
92
93     return 0;
94 }
```

Output :

```
PS C:\Users\USER\Desktop\c++> & 'c:\Users\USER\.vscode\extensions\ms-vscode.cpptools-1.28.3\Microsoft-MIEngine-Out-j2afi3lv.rnl' '--stderr=Microsoft-MIEngine-Error-dygozcm.fuk' '--pid=4364  
Added: Customer A  
Next to be served: Customer A (Deposit)  
Added: Customer B  
Added: Customer C  
Next to be served: Customer A (Deposit)  
  
--- Serving Customers ---  
Served: Customer A (Deposit)  
Served: Customer B (Withdrawal)  
Next to be served: Customer C (Loan Inquiry)  
Added: Customer D  
Served: Customer C (Loan Inquiry)  
Is the queue empty? No  
Served: Customer D (New Account)  
Is the queue empty? Yes  
PS C:\Users\USER\Desktop\c++>
```

Exercise6 :

```
1 #include <iostream>
2 using namespace std;
3
4 struct Job {
5     int id;
6     int pages;
7     Job* next;
8 };
9
10 class PrinterQueue {
11 private:
12     Job* front;
13     Job* rear;
14
15 public:
16     PrinterQueue() : front(nullptr), rear(nullptr) {}
17
18 ~PrinterQueue() {
19     while (front != nullptr) {
20         Job* temp = front;
21         front = front->next;
22         delete temp;
23     }
24 }
25
26 void enqueue(int id, int pages) {
27     Job* newJob = new Job{id, pages, nullptr};
28     if (!rear) {
29         front = rear = newJob;
30     } else {
31         rear->next = newJob;
32         rear = newJob;
33     }
34 }
35
36 Job* dequeue() {
37     if (!front) return nullptr;
38     Job* temp = front;
39     front = front->next;
40     if (!front) rear = nullptr;
41     return temp;
42 }
43
44 bool isEmpty() {
45     return front == nullptr;
46 }
```

```

47
48     void printQueue() {
49         Job* temp = front;
50         cout << "Jobs in queue:\n";
51         while (temp) {
52             cout << "Job ID: " << temp->id << ", Pages: " << temp->pages << endl;
53             temp = temp->next;
54         }
55     }
56 };
57
58 int main() {
59     PrinterQueue pq;
60
61     // Add jobs
62     pq.enqueue(1, 5);
63     pq.enqueue(2, 10);
64     pq.enqueue(3, 2);
65
66     cout << "Initial queue:\n";
67     pq.printQueue();
68     cout << "\n--- Printing Jobs ---\n";
69
70     // Print and remove jobs
71     while (!pq.isEmpty()) {
72         Job* job = pq.dequeue();
73         cout << "Printing Job ID: " << job->id << " (" << job->pages << " pages)" << endl;
74         delete job; // Remove the job once complete
75     }
76
77     cout << "\nAll jobs completed. Queue is empty.\n";
78
79     return 0;
80 }
81

```

Output :

```

PS C:\Users\USER\Desktop\c++> & 'c:\User
Microsoft-MIEngine-Out-5fmvu45y.rq2' '--s
Initial queue:
Jobs in queue:
Job ID: 1, Pages: 5
Job ID: 2, Pages: 10
Jobs in queue:
Job ID: 1, Pages: 5
Job ID: 2, Pages: 10
Job ID: 2, Pages: 10
Job ID: 3, Pages: 2

Job ID: 3, Pages: 2

--- Printing Jobs ---
--- Printing Jobs ---
Printing Job ID: 1 (5 pages)
Printing Job ID: 2 (10 pages)
Printing Job ID: 3 (2 pages)

All jobs completed. Queue is empty.
PS C:\Users\USER\Desktop\c++>

```