Name: Snehal Jayprakash Borji
UID: 2023510008
Course: F.Y.M.C.A.
Subject: DS

Practical 04

Aim: Implement Priority Queue using linked representation to serve SP-Bank.

Problem Statement: Priority Queue

1. There are 3 Queues of customers in the SP bank.
2. Bank policy is to serve, who comes first.
3. If any elder or senior citizen arrives, he is served before everyone.
4. If any VIP enters he is served before all, but senior citizens are highly respected so senior citizens are served before VIPs.

Coding:

Pract4.cpp

```
#include <iostream>

#include <queue>

#include <stdio.h>

#include <string.h>

// Structure to represent a customer

struct Customer {

std::string name;

int age;

bool isVIP;
```

```
};

// Node to represent a customer in the priority queue

struct Node {

Customer data;

Node* next;

};

class PriorityQueue {

private:

Node* front;

public:

PriorityQueue() {

front = NULL;

}

// Function to enqueue a customer based on priority

void enqueue(Customer customer) {

Node* newNode = new Node;

newNode->data = customer;

newNode->next = NULL;

if (front == NULL || isHigherPriority(customer, front->data)) {

newNode->next = front;
front = newNode;

} else {

Node* current = front;

while (current->next != NULL && !isHigherPriority(customer, current->next->data)) {

current = current->next;

}

newNode->next = current->next;

current->next = newNode;

}

}

// Function to dequeue the customer with the highest priority

Customer dequeue() {

if (isEmpty()) {

std::cerr << "Queue is empty" << std::endl;

exit(1);

}

Node* temp = front;

Customer customer = front->data;

front = front->next;

delete temp;
```

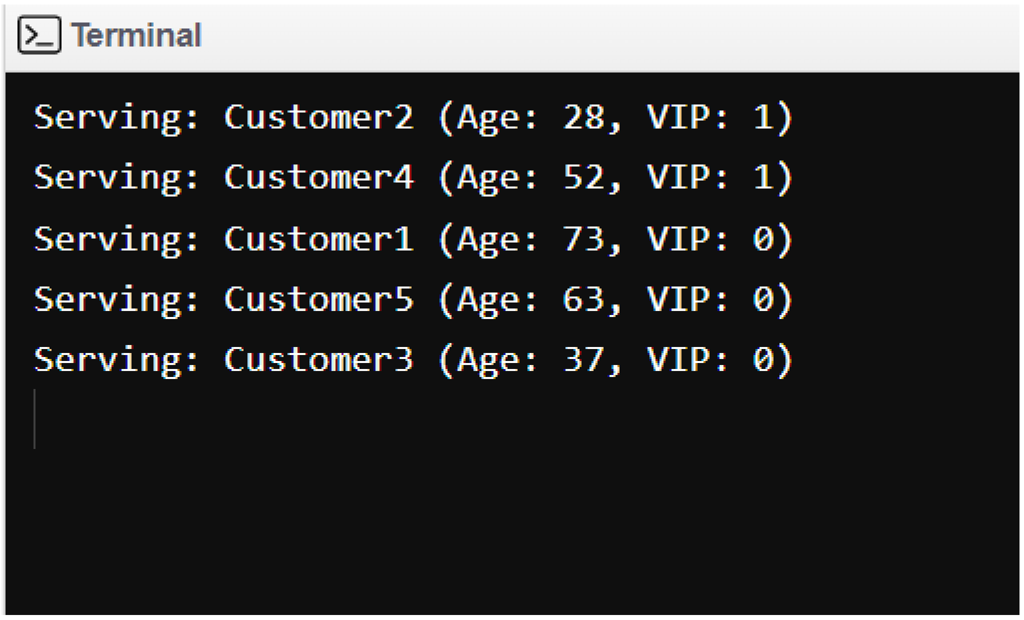
```
return customer;

}

// Function to check if the priority queue is empty

bool isEmpty() {

return front == NULL;

}

private:

// Function to check if customer1 has higher priority than customer2


bool isHigherPriority(const Customer& customer1, const Customer& customer2) {

if (customer1.isVIP && !customer2.isVIP) {

return true;

} else if (!customer1.isVIP && customer2.isVIP) {

return false;

} else {

// If both are VIP or both are not VIP, compare by age

if (customer1.age >= 60 && customer2.age < 60) {

return true; // Senior citizen has higher priority

} else if (customer1.age < 60 && customer2.age >= 60) {

return false;
} else {

return false; // In case of a tie, serve in the order they arrived

}

}

}

};

int main() {

PriorityQueue bankQueue;

// Enqueue customers

bankQueue.enqueue({"Customer1", 73, false}); // Senior Citizen

bankQueue.enqueue({"Customer2", 28, true}); // VIP

bankQueue.enqueue({"Customer3", 37, false}); // Regular

bankQueue.enqueue({"Customer4", 52, true}); // VIP

bankQueue.enqueue({"Customer5", 63, false}); // Senior Citizen


// Dequeue and serve customers in order

while (!bankQueue.isEmpty()) {

Customer servedCustomer = bankQueue.dequeue();

std::cout << "Serving: " << servedCustomer.name << " (Age: " << servedCustomer.age <<

", VIP: " << servedCustomer.isVIP << ")" << std::endl;
```

}

return 0;

}

Output:



```
Serving: Customer2 (Age: 28, VIP: 1)
Serving: Customer4 (Age: 52, VIP: 1)
Serving: Customer1 (Age: 73, VIP: 0)
Serving: Customer5 (Age: 63, VIP: 0)
Serving: Customer3 (Age: 37, VIP: 0)
```