PRN No. : 124B2B012

Name : Khairnar Atharva Anil

Title:

a) Implement a restaurant waitlist system using the queue data structure. Restaurant waitlist provide following facility:
   a. Add Party to Waitlist
   b. Seat Party
   c. Display Waitlist.

b) Implement a checkout system for a supermarket to efficiently manage customer queues during peak hours. The system should support the following operations using a circular queue data structure:
   a. Customer arrival
   b. Customer checkout
   c. Close Checkout Counter
   d. View customer.

Code:

a)

```
#include<iostream>

using namespace std;

class Node{

public:

string data;

Node *next;


public:

   Node(string data1)

   {

      data=data1;

      next=NULL;

   }

};
```

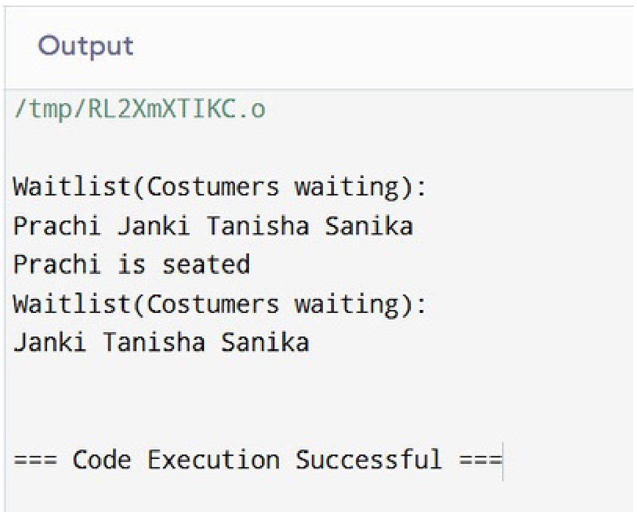
```
class Queue{
    Node *front;
    Node *rear;
    public: Queue(){
        front=rear=NULL;
    }
    void insert_wait(string data)
    {
        Node *nn=new Node(data);
        if(rear==nullptr){
            front=rear=nn;
        }
        else{
            rear->next=nn;
            rear=nn;
        }
    }

    void seat()
    {
        if(front==NULL){
            cout<<"empty!!";
    }
     Node *temp = front;
        front = front->next;
        if (front == NULL) {
            rear = NULL;
        }
        cout<<temp->data<<" is seated";
        delete temp;
    }

   void display()
   {
      if (front== NULL) {
            cout << "Queue is empty" << endl;
            return;
        }
            cout<<"\nWaitlist(Costumers waiting):"<<endl;
        Node *temp = front;
        while (temp != NULL) {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};

int main(){
    Queue q;
    q.insert_wait("Atharva");
    q.insert_wait("Aditya");
    q.insert_wait("Krishna");
    q.insert_wait("Adiyan");
    q.display(); q.seat();
    q.display();


}
```

b)

```
#include <iostream>
#include <string>

using namespace std;

class CircularQueue {
private:
    string* queue;
    int front, rear, capacity;

public:
    CircularQueue(int size) {
        capacity = size;
        queue = new string[capacity];
        front = rear = -1;
    }

     ~CircularQueue() {
        delete[] queue;
    }

    void enqueue(string customer) {
        if ((rear + 1) % capacity == front) {
            cout << "Queue is full! Cannot add customer: " << customer << endl;
            return;
        }
        if (front == -1) {
            front = rear = 0;
        } else {
            rear = (rear + 1) % capacity;
        }
        queue[rear] = customer;
        cout << "Customer " << customer << " has arrived." << endl;
    }

    void dequeue() {
        if (front == -1) {
            cout << "No customers to checkout!" << endl;
            return;
        }
        cout << "Customer " << queue[front] << " has checked out." << endl;
        if (front == rear) {
            front = rear = -1;
        } else {
            front = (front + 1) % capacity;
        }
    }

    void closeCheckout() {
        if (front == -1) {
            cout << "No customers in the queue to close the checkout." << endl;
            return;
        }
        cout << "Closing checkout. Customers remaining in the queue:" << endl;
        while (front != -1) {
            cout << queue[front] << endl;
            dequeue();
        }
    }

    void viewQueue() {
        if (front == -1) {
            cout << "The queue is empty." << endl;
            return;
        }
        cout << "Customers in the queue:" << endl;
        int i = front;
        while (true) {
            cout << queue[i] << " ";
            if (i == rear) break;
            i = (i + 1) % capacity;
        }
        cout << endl;
    }
};

int main() {
    int size;
    cout << "Enter the size of the checkout queue: ";
    cin >> size;

    CircularQueue checkoutQueue(size);

    int choice;
    string customer;
    do {
        cout << "\n1. Customer Arrival\n2. Customer Checkout\n3. Close Checkout
Counter\n4. View Customers\n5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter customer name: ";
                cin >> customer;
                checkoutQueue.enqueue(customer);
                break;
            case 2:
                checkoutQueue.dequeue();
                break;
            case 3:
                checkoutQueue.closeCheckout();
                break;
            case 4:
                checkoutQueue.viewQueue();
                break;
            case 5:
                cout << "Exiting system." << endl;
                break;
            default:
                cout << "Invalid choice! Please try again." << endl;
        }
    } while (choice != 5);
    return 0;
}
```

Output:

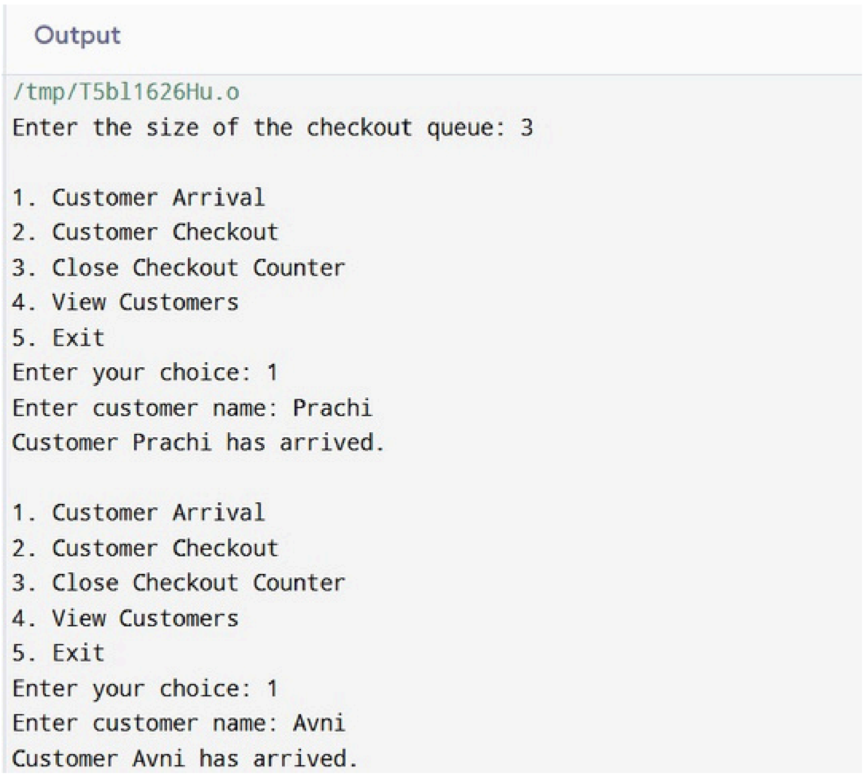
a)



Output

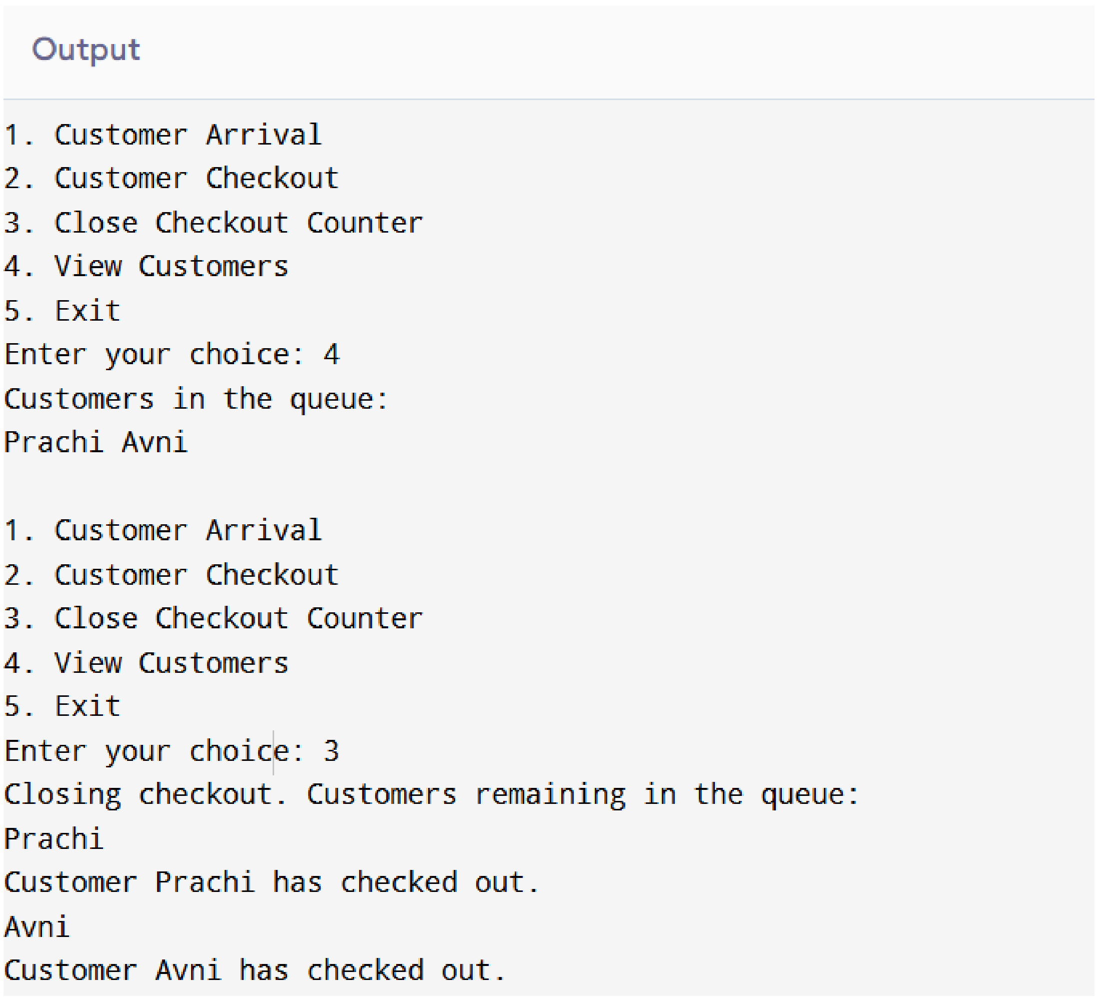
```
/tmp/RL2XmXTIKC.o

Waitlist(Costumers waiting):
Prachi Janki Tanisha Sanika
Prachi is seated
Waitlist(Costumers waiting):
Janki Tanisha Sanika


=== Code Execution Successful ===
```

b)



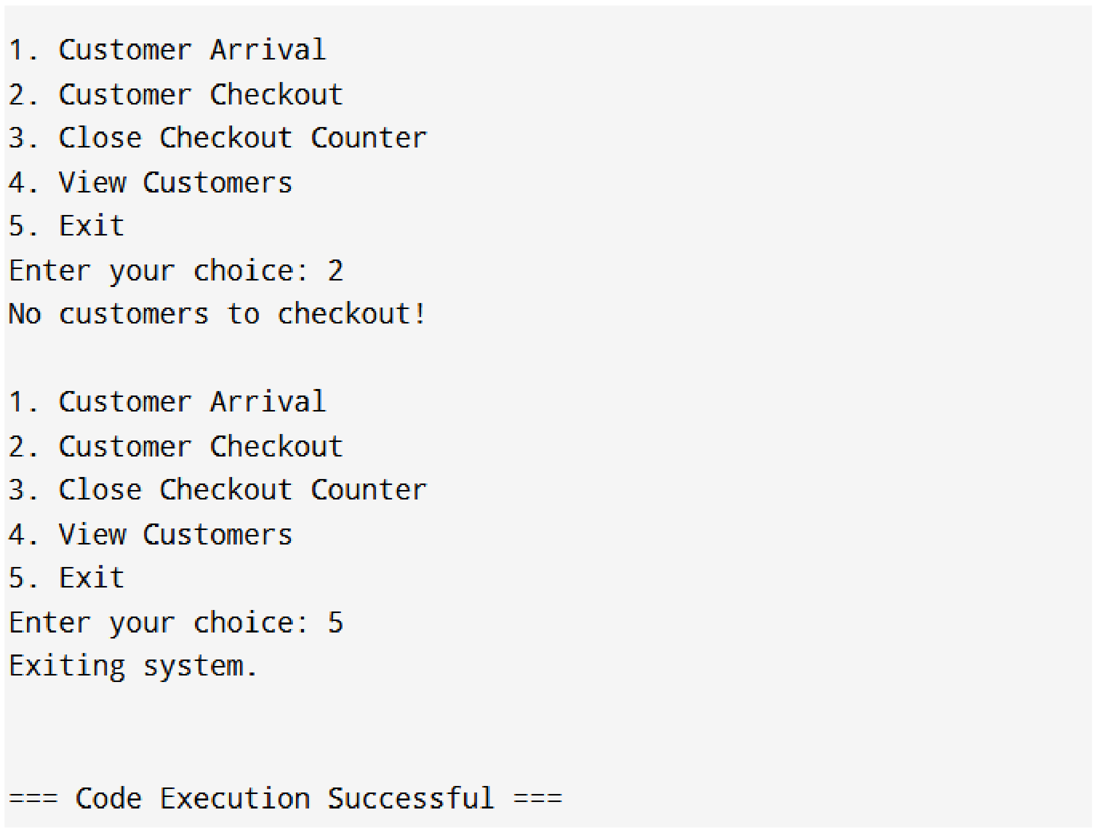
Output

```
/tmp/T5bl1626Hu.o
Enter the size of the checkout queue: 3

1. Customer Arrival
2. Customer Checkout
3. Close Checkout Counter
4. View Customers
5. Exit
Enter your choice: 1
Enter customer name: Prachi
Customer Prachi has arrived.

1. Customer Arrival
2. Customer Checkout
3. Close Checkout Counter
4. View Customers
5. Exit
Enter your choice: 1
Enter customer name: Avni
Customer Avni has arrived.
```



Output

```
1. Customer Arrival
2. Customer Checkout
3. Close Checkout Counter
4. View Customers
5. Exit
Enter your choice: 4
Customers in the queue:
Prachi Avni

1. Customer Arrival
2. Customer Checkout
3. Close Checkout Counter
4. View Customers
5. Exit
Enter your choice: 3
Closing checkout. Customers remaining in the queue:
Prachi
Customer Prachi has checked out.
Avni
Customer Avni has checked out.
```



```
1. Customer Arrival
2. Customer Checkout
3. Close Checkout Counter
4. View Customers
5. Exit
Enter your choice: 2
No customers to checkout!

1. Customer Arrival
2. Customer Checkout
3. Close Checkout Counter
4. View Customers
5. Exit
Enter your choice: 5
Exiting system.


=== Code Execution Successful ===
```