Assignment 4

Question 1

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

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

    int *queue = new int[capacity];
    int count = 0;
    int frontIndex = -1;
    int rearIndex = -1;

    while (true) {
        cout << "\n\nChoose an operation:\n";
        cout << "1. Enqueue\n";
        cout << "2. Dequeue\n";
        cout << "3. isEmpty\n";
        cout << "4. isFull\n";
        cout << "5. Display\n";
        cout << "6. Peek\n";
        cout << "7. Exit\n";2
```

```
int choice;
cin >> choice;

switch (choice) {
case 1:
    if (count < capacity) {
        int value;
        cout << "Enter the number to add to queue: ";
        cin >> value;
        if (frontIndex == -1) frontIndex = 0;
        rearIndex++;
        queue[rearIndex] = value;
        cout << "Queued " << value << " to the queue";
        count++;
    } else {
        cout << "Queue is full!";
    }
    break;

case 2:
    if (count != 0) {
        int removed = queue[frontIndex];
        cout << "Removed " << removed << " from the queue";
        frontIndex++;
        count--;
        if (count == 0) {
```

```
            frontIndex = -1;
            rearIndex = -1;
        }
    } else {
        cout << "The queue is already empty!";
    }
    break;

case 3:
    if (count == 0) {
        cout << "The queue is empty";
    } else {
        cout << "The queue is not empty";
    }
    break;

case 4:
    if (count == capacity) {
        cout << "The queue is full";
    } else {
        cout << "The queue is not full";
    }
    break;

case 5:
    if (count == 0) {
```

```
        cout << "Queue is empty!";
    } else {
        cout << "Queue elements: ";
        for (int i = frontIndex; i <= rearIndex; i++) {
            cout << queue[i] << " ";
        }
    }
    break;

case 6:
    if (count == 0) {
        cout << "Queue is empty!";
    } else {
        cout << "Front element: " << queue[frontIndex];
    }
    break;

case 7:
    cout << "Exiting!";
    delete[] queue;
    return 0;

default:
    cout << "Invalid choice. Try again!";
    break;
}
```

```
    }
}
```

```
4. isFull
5. Display
6. Peek
7. Exit
5
Queue is empty!

Choose an operation:
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
```

Question 2

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

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

    int *queue = new int[capacity];
    int count = 0;
    int frontIndex = -1;
    int rearIndex = -1;

    while (true) {
```

```
cout << "\n\nChoose an operation:\n";
cout << "1. Enqueue\n";
cout << "2. Dequeue\n";
cout << "3. isEmpty\n";
cout << "4. isFull\n";
cout << "5. Display\n";
cout << "6. Peek\n";
cout << "7. Exit\n";

int choice;
cin >> choice;

switch (choice) {
case 1:
    if (count < capacity) {
        if (frontIndex == -1) frontIndex = 0;
        int value;
        cout << "Enter the number to add to queue: ";
        cin >> value;
        rearIndex = (rearIndex + 1) % capacity;
        queue[rearIndex] = value;
        cout << "Queued " << value << " to the queue";
        count++;
    } else {
        cout << "Queue is full!";
    }
```

```
    break;

case 2:
    if (count != 0) {
        int removed = queue[frontIndex];
        frontIndex = (frontIndex + 1) % capacity;
        count--;
        cout << "Removed " << removed << " from the queue";
        if (count == 0) {
            frontIndex = -1;
            rearIndex = -1;
        }
    } else {
        cout << "The queue is already empty!";
    }
    break;

case 3:
    if (count == 0) {
        cout << "The queue is empty";
    } else {
        cout << "The queue is not empty";
    }
    break;

case 4:
```

```
    if (count == capacity) {
        cout << "The queue is full";
    } else {
        cout << "The queue is not full";
    }
    break;

case 5:
    if (count == 0) {
        cout << "Queue is empty!";
    } else {
        cout << "Queue elements: ";
        for (int i = 0; i < count; i++) {
            cout << queue[(frontIndex + i) % capacity] << " ";
        }
    }
    break;

case 6:
    if (count == 0) {
        cout << "Queue is empty!";
    } else {
        cout << "Front element: " << queue[frontIndex];
    }
    break;
```

```
        case 7:
            cout << "Exiting!";
            delete[] queue;
            return 0;

        default:
            cout << "Invalid choice. Try again!";
            break;
        }
    }
}
```

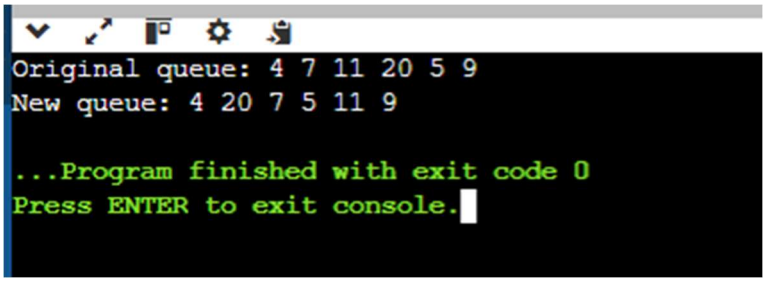
```
Enter the size of the queue: 4


Choose an operation:
1. Enqueue
2. Dequeue
3. isEmpty
4. isFull
5. Display
6. Peek
7. Exit
1
Enter the number to add to queue: 2
Queued 2 to the queue
```

Question 3

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

int main() {
    queue<int> q;
```

```
int arr[] = {4, 7, 11, 20, 5, 9};
int n = sizeof(arr) / sizeof(arr[0]);

for (int i = 0; i < n; i++) {
    q.push(arr[i]);
}

queue<int> temp = q;
cout << "Original queue: ";
for (int i = 0; i < n; i++) {
    cout << temp.front() << " ";
    temp.pop();
}

temp = q;
queue<int> secondHalf;
for (int i = 0; i < n / 2; i++) {
    temp.pop();
}
secondHalf = temp;

queue<int> firstHalf = q;
queue<int> result;
for (int i = 0; i < n / 2; i++) {
    result.push(firstHalf.front());
    firstHalf.pop();
```

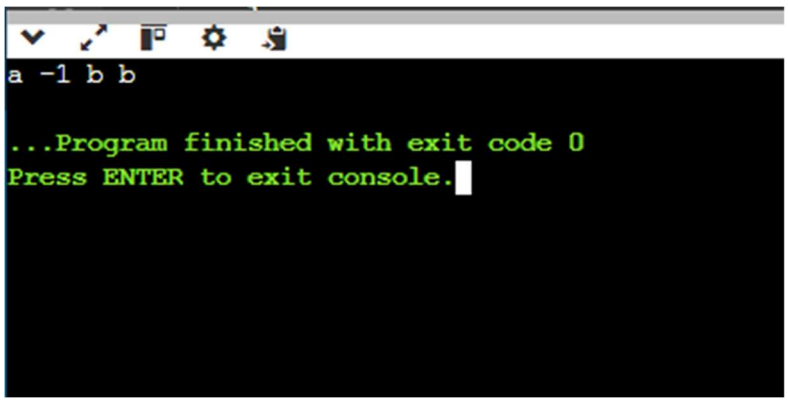
```
        result.push(secondHalf.front());
        secondHalf.pop();
    }

    cout << "\nNew queue: ";
    while (!result.empty()) {
        cout << result.front() << " ";
        result.pop();
    }

    return 0;
}
```



Question 4

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

int main() {
    string input = "aabc";
    int charCount[26] = {0};
```

```
    queue<char> qChars;

    for (char ch : input) {
        charCount[ch - 'a']++;
        qChars.push(ch);

        while (!qChars.empty() && charCount[qChars.front() - 'a'] > 1) {
            qChars.pop();
        }

        if (qChars.empty()) {
            cout << "-1 ";
        } else {
            cout << qChars.front() << " ";
        }
    }

    return 0;
}
```



Question 5

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

queue<int> q, q1;

void push(int value) {
    q1.push(value);
    while (!q.empty()) {
        q1.push(q.front());
        q.pop();
    }
    swap(q, q1);
}

void pop() {
    if (q.empty()) {
        cout << "Stack is empty!\n";
        return;
    }
    cout << "Popped: " << q.front() << endl;
    q.pop();
}
```

```
int top() {
    if (q.empty()) {
        cout << "Stack is empty!\n";
        return -1;
    }
    return q.front();
}

bool isEmpty() {
    return q.empty();
}

int mainA() {
    cout << "--- Using two queues ---" << endl;
    push(10);
    push(20);
    push(30);

    cout << "Top: " << top() << endl;
    pop();
    cout << "Top: " << top() << endl;
    pop();
    pop();
    pop();
    return 0;
}
```

```
queue<int> qSingle;

void pushSingle(int value) {
    int size = qSingle.size();
    qSingle.push(value);
    for (int i = 0; i < size; i++) {
        qSingle.push(qSingle.front());
        qSingle.pop();
    }
}

void popSingle() {
    if (qSingle.empty()) {
        cout << "Stack is empty!\n";
        return;
    }
    cout << "Popped: " << qSingle.front() << endl;
    qSingle.pop();
}

int topSingle() {
    if (qSingle.empty()) {
        cout << "Stack is empty!\n";
        return -1;
    }
```

```
    return qSingle.front();
}

bool isEmptySingle() {
    return qSingle.empty();
}

int main() {
    mainA();

    cout << "\n--- Using one queue ---" << endl;
    pushSingle(10);
    pushSingle(20);
    pushSingle(30);

    cout << "Top: " << topSingle() << endl;
    popSingle();
    cout << "Top: " << topSingle() << endl;
    popSingle();
    popSingle();
    popSingle();
}
```

```
--- Using two queues ---
Top: 30
Popped: 30
Top: 20
Popped: 20
Popped: 10
Stack is empty!

--- Using one queue ---
Top: 30
Popped: 30
Top: 20
Popped: 20
Popped: 10
Stack is empty!
```