AQ1.cpp

```
#include <iostream>

using namespace std;


int *arr;

int SIZE;

int front=-1, rear=-1;


bool isEmpty()

{

   return (front==-1 && rear==-1);

}


bool isFull()

{

   return (rear==SIZE-1);

}


void enqueue(int num)

{

   if (isFull())

   {

      cout<<"Queue is Full!"<<endl;

      return;

   }

   if (isEmpty())

   {
```

```
        front=0;
    }
    rear++;
    arr[rear]=num;
    cout<<num<<" inserted into queue"<<endl;
}

void dequeue()
{
    if (isEmpty())
    {
        cout<<"Queue is Empty!"<<endl;
        return;
    }
    cout<<arr[front]<<" removed from queue"<<endl;
    if (front==rear)
    {
        front=rear=-1;
    }
    else
    {
        front++;
    }
}

void peek()
{
    if (isEmpty())
```

```
    {
        cout<<"Queue is Empty!"<<endl;
    }
    else
    {
        cout<<"Front element is: "<<arr[front]<<endl;
    }
}

void display()
{
    if (isEmpty())
    {
        cout<<"Queue is Empty!"<<endl;
    }
    else
    {
        cout<<"Queue elements: ";
        for (int i=front; i <= rear; i++)
        {
            cout<<arr[i]<<" ";
        }
        cout<<endl;
    }
}

int main()
{
```

```
int choice, num;
cout<<"Enter size of queue: ";
cin >> SIZE;
arr=new int[SIZE];

do
{
   cout<<"Queue Menu\n";
   cout<<"1. Enqueue\n";
   cout<<"2. Dequeue\n";
   cout<<"3. Peek\n";
   cout<<"4. Display\n";
   cout<<"5. Check if Empty\n";
   cout<<"6. Check if Full\n";
   cout<<"7. Exit\n";
   cout<<"Enter your choice: ";
   cin >> choice;

   switch (choice) {
   case 1:
      cout<<"Enter number to insert: ";
      cin>>num;
      enqueue(num);
      break;
   case 2:
      dequeue();
      break;
   case 3:
```

```
            peek();
            break;
        case 4:
            display();
            break;
        case 5:
            if (isEmpty())
                cout<<"Queue is Empty!"<<endl;
            else
                cout<<"Queue is NOT Empty!"<<endl;
            break;
        case 6:
            if (isFull())
                cout<<"Queue is Full!"<<endl;
            else
                cout<<"Queue is NOT Full!"<<endl;
            break;
        case 7:
            cout<<"Exited"<<endl;
            break;
        default:
            cout<<"Invalid choice, try again!"<<endl;
        }
    } while (choice != 7);
    delete[] arr;
    return 0;
}
```

```
Enter size of queue: 3
Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 1
Enter number to insert: 22
22 inserted into queue
Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 1
Enter number to insert: 33
33 inserted into queue
Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 1
Enter number to insert: 44
44 inserted into queue
Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 4
Queue elements: 22 33 44
Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 7
Exited
PS F:\Work\SEM3\DSA\4>
```

AQ2.cpp

```
#include <iostream>

using namespace std;


int *arr;

int SIZE;

int front=-1, rear=-1;


bool isEmpty()

{

   return (front==-1 && rear==-1);

}


bool isFull()

{

   return ((rear + 1) % SIZE==front);

}


void enqueue(int num)

{

   if (isFull())

   {

      cout<<"Queue is Full!"<<endl;

      return;

   }

   if (isEmpty())

   {
```

```
        front=rear=0;
    }
    else
    {
        rear=(rear+1) % SIZE;
    }
    arr[rear]=num;
    cout<<num<<" inserted into queue"<<endl;
}

void dequeue()
{
    if (isEmpty())
    {
        cout<<"Queue is Empty!"<<endl;
        return;
    }
    cout<<arr[front]<<" removed from queue"<<endl;
    if (front==rear)
    {
        front=rear=-1;
    }
    else
    {
        front=(front + 1) % SIZE;
    }
}
```

```
void peek()
{
   if (isEmpty())
   {
      cout<<"Queue is Empty!"<<endl;
   }
   else
   {
      cout<<"Front element is: "<<arr[front]<<endl;
   }
}

void display()
{
   if (isEmpty())
   {
      cout<<"Queue is Empty!"<<endl;
   }
   else
   {
      cout<<"Queue elements: ";
      int i=front;
      while (true) {
         cout<<arr[i]<<" ";
         if (i==rear) break;
         i=(i + 1) % SIZE;
      }
      cout<<endl;
```

```
    }
}

int main()
{
    int choice, num;
    cout<<"Enter size of queue: ";
    cin >> SIZE;
    arr=new int[SIZE];
    do {
        cout<<"Circular Queue Menu\n";
        cout<<"1. Enqueue\n";
        cout<<"2. Dequeue\n";
        cout<<"3. Peek\n";
        cout<<"4. Display\n";
        cout<<"5. Check if Empty\n";
        cout<<"6. Check if Full\n";
        cout<<"7. Exit\n";
        cout<<"Enter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout<<"Enter number to insert: ";
            cin >> num;
            enqueue(num);
            break;
        case 2:
```

```
            dequeue();
            break;
        case 3:
            peek();
            break;
        case 4:
            display();
            break;
        case 5:
            if (isEmpty())
                cout<<"Queue is Empty!"<<endl;
            else
                cout<<"Queue is NOT Empty!"<<endl;
            break;
        case 6:
            if (isFull())
                cout<<"Queue is Full!"<<endl;
            else
                cout<<"Queue is NOT Full!"<<endl;
            break;
        case 7:
            cout<<"Exited"<<endl;
            break;
        default:
            cout<<"Invalid choice, try again!"<<endl;
        }
    } while (choice != 7);
```

```
    delete[] arr;

    return 0;
}
```

```
Enter size of queue: 3
Circular Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 1
Enter number to insert: 11
11 inserted into queue
Circular Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 1
Enter number to insert: 22
22 inserted into queue
Circular Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 1
Enter number to insert: 33
33 inserted into queue
Circular Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 4
Queue elements: 11 22 33
Circular Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 7
Exited
PS F:\Work\SEM3\DSA\4>
```

AQ3.cpp

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

void interleaveQueue(queue<int> &q)
{
    int n=q.size();
    if(n%2!=0)
    {
        cout<<"Queue has odd number of elements, cannot interleave!"<<endl;
        return;
    }

    int half=n/2;
    queue<int> firstHalf;

    for(int i=0; i<half; i++)
    {
        firstHalf.push(q.front());
        q.pop();
    }

    while(!firstHalf.empty())
    {
        q.push(firstHalf.front());
        firstHalf.pop();
```

```
        q.push(q.front());
        q.pop();
    }
}

void display(queue<int> q)
{
    while(!q.empty())
    {
        cout<<q.front()<<" ";
        q.pop();
    }
    cout<<endl;
}

int main()
{
    queue<int> q;
    int n, num;

    cout<<"Enter number of elements: ";
    cin >> n;

    cout<<"Enter elements: ";
    for(int i=0; i<n; i++)
    {
        cin >> num;
        q.push(num);
```

```
 }

    cout<<"Original Queue: ";
    display(q);

    interleaveQueue(q);

    cout<<"Interleaved Queue: ";
    display(q);

    return 0;
}
```

```
Enter number of elements: 6
Enter elements: 11
22
33
44
55
66
Original Queue: 11 22 33 44 55 66
Interleaved Queue: 11 44 22 55 33 66
PS F:\Work\SEM3\DSA\4>
```

AQ4.cpp

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

int main() {
    string str;
    cout << "Enter string: ";
```

```
    cin >> str;

    queue<char> q;
    int freq[26] = {0};

    for (int i=0; i<str.length();i++)
    {
        char ch=str[i];
        freq[ch-'a']++;
        q.push(ch);

        while (!q.empty() && freq[q.front()-'a']>1)
        {
            q.pop();
        }
        if (q.empty())
            cout<<-1<<" ";
        else
            cout<<q.front()<<" ";
    }
    return 0;
}
```

```
Enter string: aabc
a -1 b b
PS F:\Work\SEM3\DSA\4>
```

AQ5.cpp

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

queue<int> q1, q2;
queue<int> q;

void pushTwoQueues(int x)
{
   q2.push(x);
   while (!q1.empty())
   {
      q2.push(q1.front());
      q1.pop();
   }
   swap(q1, q2);
}

void popTwoQueues()
{
   if(q1.empty())
   {
      cout<<"Stack is empty"<<endl;
      return;
   }
   cout<<q1.front()<<" popped from stack"<<endl;
```

```
    q1.pop();
}

int topTwoQueues()
{
    if(q1.empty())
    {
        cout<<"Stack is empty"<<endl;
        return -1;
    }
    return q1.front();
}

void displayTwoQueues()
{
    if(q1.empty())
    {
        cout<<"Stack is empty"<<endl;
        return;
    }
    queue<int> temp=q1;
    cout<<"Stack elements: ";
    while (!temp.empty())
    {
        cout<<temp.front()<<" ";
        temp.pop();
    }
    cout<<endl;
```

```
}

void pushOneQueue(int x)
{
   int sz=q.size();
   q.push(x);
   for (int i=0; i<sz; i++)
   {
      q.push(q.front());
      q.pop();
   }
}

void popOneQueue()
{
   if(q.empty())
   {
      cout<<"Stack is empty"<<endl;
      return;
   }
   cout<<q.front()<<" popped from stack"<<endl;
   q.pop();
}

int topOneQueue()
{
   if(q.empty())
   {
```

```
        cout<<"Stack is empty"<<endl;
        return -1;
    }
    return q.front();
}

void displayOneQueue()
{
    if(q.empty())
    {
        cout<<"Stack is empty"<<endl;
        return;
    }
    queue<int> temp=q;
    cout<<"Stack elements: ";
    while (!temp.empty())
    {
        cout<<temp.front()<<" ";
        temp.pop();
    }
    cout<<endl;
}

int main()
{
    int method, choice, num;
    cout<<"Choose Stack Implementation:\n1. Two Queues\n2. One Queue\nEnter choice: ";
    cin >> method;
```

```
if(method==1)
{
   do {
      cout<<"Stack (Two Queues) Menu\n";
      cout<<"1. Push\n2. Pop\n3. Top\n4. Display\n5. Exit\nEnter choice: ";
      cin >> choice;
      switch (choice) {
         case 1:
            cout<<"Enter element: ";
            cin >> num;
            pushTwoQueues(num);
            break;
         case 2:
            popTwoQueues();
            break;
         case 3:
            cout<<"Top element: "<<topTwoQueues()<<endl;
            break;
         case 4:
            displayTwoQueues();
            break;
         case 5:
            cout<<"Exited"<<endl;
            break;
         default:
            cout<<"Invalid choice"<<endl;
      }
```

```
    } while (choice != 5);
}
else if(method == 2)
{
    do {
        cout<<"Stack(One Queue)Menu\n";
        cout<<"1. Push\n2. Pop\n3. Top\n4. Display\n5. Exit\nEnter choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                cout<<"Enter element: ";
                cin >> num;
                pushOneQueue(num);
                break;
            case 2:
                popOneQueue();
                break;
            case 3:
                cout<<"Top element: "<<topOneQueue()<<endl;
                break;
            case 4:
                displayOneQueue();
                break;
            case 5:
                cout<<"Exited"<<endl;
                break;
            default:
                cout<<"Invalid choice"<<endl;
```

```
        }
    } while (choice != 5);
  }
  else
  {
    cout<<"Invalid method choice"<<endl;
  }
  return 0;
}
```

```
Choose Stack Implementation:
1. Two Queues
2. One Queue
Enter choice: 1
Stack (Two Queues) Menu
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter choice: 1
Enter element: 44
Stack (Two Queues) Menu
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter choice: 1
Enter element: 55
Stack (Two Queues) Menu
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter choice: 1
Enter element: 66
Stack (Two Queues) Menu
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter choice: 4
Stack elements: 66 55 44
Stack (Two Queues) Menu
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter choice: 5
Exited
PS F:\Work\SEM3\DSA\4>
```

```
Choose Stack Implementation:
1. Two Queues
2. One Queue
Enter choice: 2
Stack(One Queue)Menu
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter choice: 1
Enter element: 22
Stack(One Queue)Menu
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter choice: 1
Enter element: 44
Stack(One Queue)Menu
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter choice: 1
Enter element: 66
Stack(One Queue)Menu
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter choice: 3
Top element: 66
Stack(One Queue)Menu
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter choice: 5
Exited
PS F:\Work\SEM3\DSA\4>
```