Inclusive array based implementation:

```
#include <iostream>
#define MAX_SIZE 5
using namespace std;

class Queue {
private:
int myqueue[MAX_SIZE], front, rear;

public:
Queue(){
front = -1;
rear = -1;
    }

boolisFull(){
if(front == 0 && rear == MAX_SIZE - 1){
return true;
        }
return false;
    }

boolisEmpty(){
if(front == -1) return true;
else return false;
    }
```

```
void enQueue(int value){
if(isFull()){
cout << endl<< "Queue is full!!";
        } else {
if(front == -1) front = 0;
rear++;
myqueue[rear] = value;
cout << value << " ";
        }
    }
int deQueue(){
int value;
if(isEmpty()){
cout << "Queue is empty!!" << endl; return(-1); } else { value = myqueue[front];
if(front >= rear){      //only one element in queue
front = -1;
rear = -1;
            }
else {
front++;
            }
cout << endl << "Deleted => " << value << " from myqueue";
return(value);
        }
    }
```

```
void displayQueue()
    {
int i;
if(isEmpty()) {
cout << endl << "Queue is Empty!!" << endl;
        }
else {
cout << endl << "Front = " << front;
cout << endl << "Queue elements : ";
for(i=front; i<=rear; i++)
cout << myqueue[i] << "\t";
cout << endl << "Rear = " << rear << endl;
        }
    }
};
int main()
{
   Queue myq;
myq.deQueue();      //deQueue

cout<<"Queue created:"<<endl;
myq.enQueue(10); myq.enQueue(20); myq.enQueue(30); myq.enQueue(40); myq.enQueue(50);
//enqueue 60 => queue is full
myq.enQueue(60);
myq.displayQueue();
      //deQueue =>removes 10
myq.deQueue();
    //queue after dequeue
myq.displayQueue();
return 0;
}
```

Another implementation

```
#include<iostream>

using namespace std;

class Queue {
  private:
    int front;
  int rear;
  int arr[5];

  public:
    Queue() {
      front = -1;
      rear = -1;
      for (int i = 0; i < 5; i++) {
        arr[i] = 0;
      }
    }
  bool isEmpty() {
    if (front == -1 && rear == -1)
      return true;
    else
      return false;
  }
```

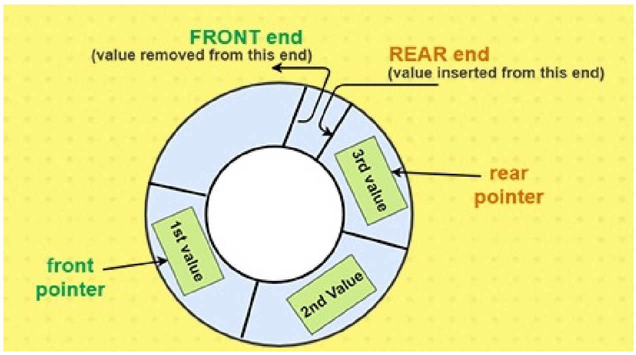
```
bool isFull() {
   if (rear == 4)
     return true;
   else
     return false;
 }
 void enqueue(int val) {
   if (isFull()) {
     cout << "Queue full" << endl;
     return;
   } else if (isEmpty()) {
     rear = 0;
     front = 0;
     arr[rear] = val;
   } else {
     rear++;
     arr[rear] = val;
   }

 }
```

```
 int dequeue() {
   int x = 0;
   if (isEmpty()) {
     cout << "Queue is Empty" << endl;
     return x;
   } else if (rear == front) {
     x = arr[rear];
     rear = -1;
     front = -1;
     return x;
   } else {
     cout << "front value: " << front << endl;
     x = arr[front];
     arr[front] = 0;
     front++;
     return x;
   }
 }

 int count() {
   return (rear - front + 1);
 }
 void display() {
    cout << "All values in the Queue are - " << endl;
    for (int i = 0; i < 5; i++) {
      cout << arr[i] << "  ";
    }
  }

};

int main() {
  Queue q1;
  int value, option;

  do {
    cout << "\n\nWhat operation do you want to perform? Select Option number.
Enter 0 to exit." << endl;
    cout << "1. Enqueue()" << endl;
    cout << "2. Dequeue()" << endl;
    cout << "3. isEmpty()" << endl;
    cout << "4. isFull()" << endl;
    cout << "5. count()" << endl;
    cout << "6. display()" << endl;
    cout << "7. Clear Screen" << endl << endl;
```

```
  cin >> option;

    switch (option) {
    case 0:
      break;
    case 1:
      cout << "Enqueue Operation \nEnter an item to Enqueue in the Queue" <<
endl;
      cin >> value;
      q1.enqueue(value);
      break;
    case 2:
      cout << "Dequeue Operation \nDequeued Value : " << q1.dequeue() << endl;
      break;
    case 3:
      if (q1.isEmpty())
        cout << "Queue is Empty" << endl;
      else
        cout << "Queue is not Empty" << endl;
      break;
```

```
 case 5:
      cout << "Count Operation \nCount of items in Queue : " << q1.count() <<
endl;
      break;
    case 6:
      cout << "Display Function Called - " << endl;
      q1.display();
      break;
    case 7:
      system("cls");
      break;
    default:
      cout << "Enter Proper Option number " << endl;
    }

  } while (option != 0);

  return 0;
}
```

Wrap Around Technique:(Circular Queue):



```
#include<iostream>

using namespace std;

class CircularQueue {
  private:
    int front;
  int rear;
  int arr[5];
  int itemCount;

  public:
    CircularQueue() {
      itemCount = 0;
      front = -1;
      rear = -1;
      for (int i = 0; i < 5; i++) {
        arr[i] = 0;
      }
    }
  bool isEmpty() {
    if (front == -1 && rear == -1)
      return true;
    else
      return false;
  }
```

```
bool isFull() {
    if ((rear + 1) % 5 == front)
      return true;
    else
      return false;
  }
  void enqueue(int val) {
    if (isFull()) {
      cout << "Queue full" << endl;
      return;
    } else if (isEmpty()) {
      rear = 0;
      front = 0;
      arr[rear] = val;
    } else {
      rear = (rear + 1) % 5;
      arr[rear] = val;
    }
   itemCount++;
  }
```

```
 int dequeue() {
    int x = 0;
    if (isEmpty()) {
      cout << "Queue is Empty" << endl;
      return x;
    } else if (rear == front) {
      x = arr[rear];
      rear = -1;
      front = -1;
      itemCount--;
      return x;
    } else {
      cout << "front value: " << front << endl;
      x = arr[front];
      arr[front] = 0;
      front = (front + 1) % 5;
      itemCount--;
      return x;
    }
  }
```

```
int count() {
    return (itemCount);
  }
  void display() {
    cout << "All values in the Queue are - " << endl;
    for (int i = 0; i < 5; i++) {
      cout << arr[i] << "   ";
    }
  }
};
int main() {
  CircularQueue q1;
  int value, option;
```

```
do {
    cout << "\n\nWhat operation do you want to perform? Select Option number.
Enter 0 to exit." << endl;
    cout << "1. Enqueue()" << endl;
    cout << "2. Dequeue()" << endl;
    cout << "3. isEmpty()" << endl;
    cout << "4. isFull()" << endl;
    cout << "5. count()" << endl;
    cout << "6. display()" << endl;
    cout << "7. Clear Screen" << endl << endl;
    cin >> option;
```

```
switch (option) {
    case 0:
      break;
    case 1:
      cout << "Enqueue Operation \nEnter an item to Enqueue in the Queue" <<
endl;
      cin >> value;
      q1.enqueue(value);
      break;
    case 2:
      cout << "Dequeue Operation \nDequeued Value : " << q1.dequeue() << endl;
      break;
    case 3:
      if (q1.isEmpty())
        cout << "Queue is Empty" << endl;
      else
        cout << "Queue is not Empty" << endl;
      break;
    case 4:
      if (q1.isFull())
        cout << "Queue is Full" << endl;
      else
        cout << "Queue is not Full" << endl;
      break;
```

```
case 5:
      cout << "Count Operation \nCount of items in Queue : " << q1.count() <<
endl;
      break;
    case 6:
      cout << "Display Function Called - " << endl;
      q1.display();
      break;
    case 7:
      system("cls");
      break;
    default:
      cout << "Enter Proper Option number " << endl;
    }

  } while (option != 0);

  return 0;
}
```

Priority Queue:

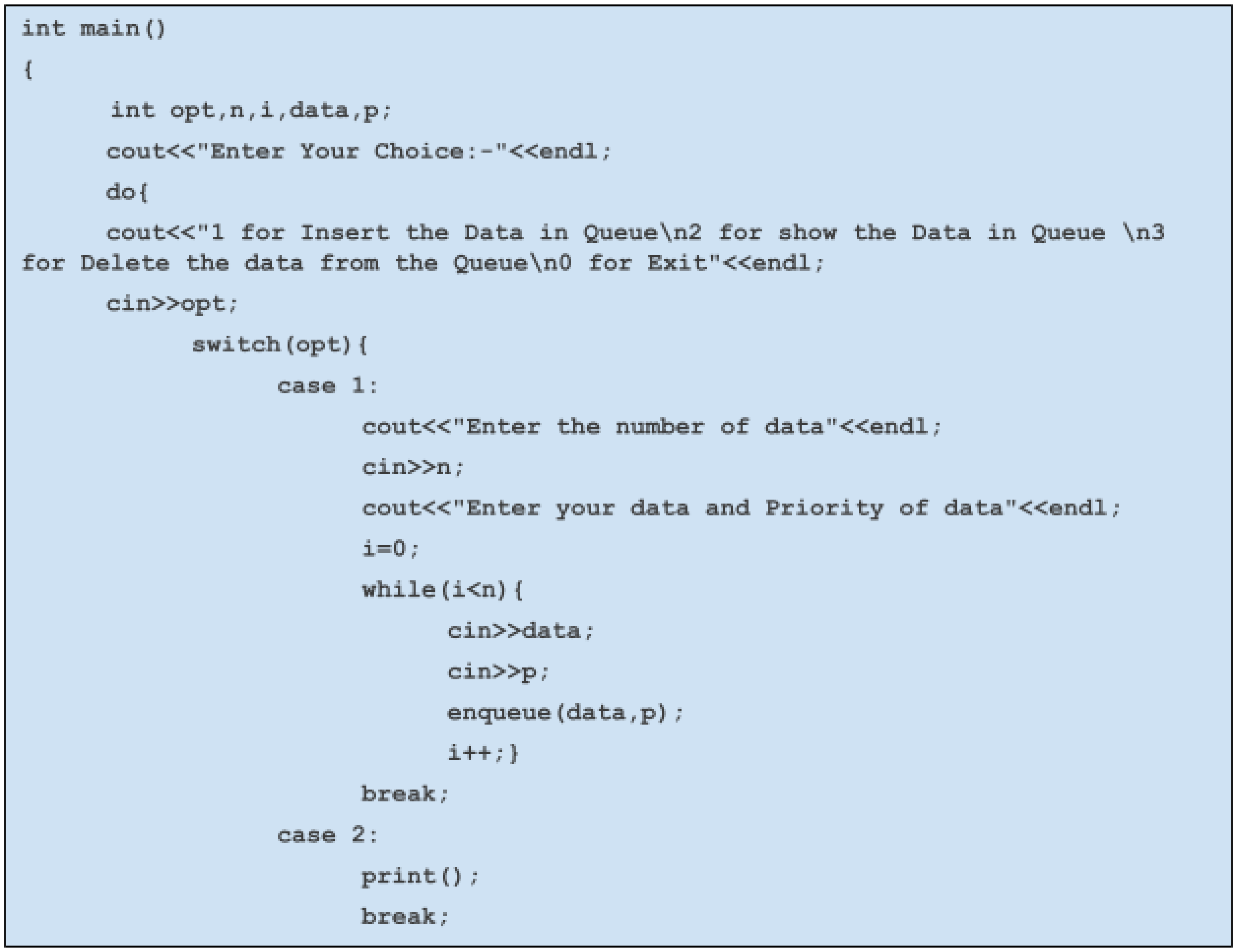
Operation	Priority Queue	Return value
Insert(G)	G	
Insert(O)	G O	
Insert(M)	G O M	
deleteHighestPriority()	G M	O
Insert(A)	G M A	
deleteHighestPriority()	G A	M

C++ Program to Implement Max Priority Queue (using Ordered Array)

```
#include<iostream>
#define N 20
using namespace std;
int Q[N],Pr[N];
int r = -1,f = -1;
void enqueue(int data,int p)//Enqueue function to insert data and its priority in queue
{
        int i;
        if((f==0)&&(r==N-1)) //Check if Queue is full
                cout<<"Queue is full";
        else
        {
                if(f==-1)//if Queue is empty
                {
                        f = r = 0;
                        Q[r] = data;
                        Pr[r] = p;
}
            else if(r == N-1)//if there there is some elemets in Queue
            {
                  for(i=f;i<=r;i++) {
                                  Q[i-f] = Q[i];
                                  Pr[i-f] = Pr[i];
                                  r = r-f;
                                  f = 0;
                                  for(i = r;i>f;i--)
                        {
                              if(p>Pr[i])
                              {
                                    Q[i+1] = Q[i];
                                    Pr[i+1] = Pr[i];
                              }
                              else
                                    break;
                              Q[i+1] = data;
                              Pr[i+1] = p;
                              r++;
                        }
                  }
            }
```

```
else
            {
                  for(i = r;i>=f;i--)
                  {
                        if(p>Pr[i])
                        {
                              Q[i+1] = Q[i];
                              Pr[i+1] = Pr[i];
                        }
                        else
                              break;
                  }
                  Q[i+1] = data;
                  Pr[i+1] = p;
                  r++;
            }
      }
}
```

```
void print() //print the data of Queue
{
int i;
      for(i=f;i<=r;i++)
      {
            cout<<"Element ="<<Q[i]<<"Priority = "<<Pr[i]<<endl;
      }
}
int dequeue() //remove the data from front
{
      if(f == -1)
      {
      cout<<"Queue is Empty";
      }
      else
      {
      cout<<"deleted Element ="<<Q[f]<<endl;
      cout<<"Its Priority = "<<Pr[f]<<endl;
            if(f==r)
                  f = r = -1;
            else
                  f++;  }
}
```



```
int main()
{
      int opt,n,i,data,p;
      cout<<"Enter Your Choice:-"<<endl;
      do{
      cout<<"1 for Insert the Data in Queue\n2 for show the Data in Queue \n3
for Delete the data from the Queue\n0 for Exit"<<endl;
      cin>>opt;
            switch(opt){
                  case 1:
                        cout<<"Enter the number of data"<<endl;
                        cin>>n;
                        cout<<"Enter your data and Priority of data"<<endl;
                        i=0;
                        while(i<n){
                              cin>>data;
                              cin>>p;
                              enqueue(data,p);
                              i++;}
                        break;
                  case 2:
                        print();
                        break;
```

Question: 07 Palindrome:

```
#include <iostream>
using namespace std;
class node{
     public:
     char data;
     node *next;

     node(){
          data=0;
          next=NULL;
     }
     node(char d){
          data=d;
          next=NULL;
     }
};
class queue{
     node *head;

     public:
          queue(){
               head=NULL;
          }
          void enqueueCharacter(char s){
               if(head==NULL){
                    head=new node(s);
                    return;
               }
               node *temp=head;
               head=new node(s);
               head->next=temp;
          }
          void dequeueCharacter(){
               node *temp=head;
               head=head->next;
               delete temp;
          }
          void print(){
               for(node *temp=head;temp!=NULL;temp=temp->next){
                    cout << temp->data << " " ;
               }
               cout << endl;
          }
          char front(){
               return head->data;
          }
};
```

```
class stack{
    node *head;

    public:
        stack(){
            head=NULL;
        }
        void push(char d){
            if(head==NULL){
                head=new node(d);
                return;
            }
            node *temp=head;
            while(temp->next!=NULL){
                temp=temp->next;
            }
            temp->next=new node(d);
        }
        void popCharacter(){
            node *temp=head;
            head=head->next;
            delete temp;
        }
        void print(){
            for(node *temp=head;temp!=NULL;temp=temp->next){
                cout << temp->data << " " ;
            }
            cout << endl;
        }
        char top(){
            return head->data;
        }
};
bool isPalindrome(stack s,queue q,string st){
     for(int i=0;i<st.length();i++){
          if(q.front()!=s.top()){
               return false;
          }
     }
     return true;
}
int main(){
     stack s;
     queue q;
     string st;
     cout << "Enter string(is composed of lowercase English
letters): ";
     cin >> st;
     for(int i=0;i<st.length();i++){
          s.push(st[i]);
          q.enqueueCharacter(st[i]);
     }
     if(isPalindrome(s,q,st)){
          cout << "The word," << st << ", is a palindrome";
     } else {
          cout << "The word," << st << ", is not a
palindrome";
     }


}
```