ASSIGNMENT NO. 3 (QUEUE)

1) Write a program to implement linear queue by using array. (Static Implementation of queue)

->

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

class  queue {
     int item[max], front, rear;
     public:
       void   create(queue*);
       void   isempty(queue*);
       void   isfull(queue*);
       void   insert(queue*, int);
       int    remove(queue*);
       void   display(queue*);
};

void   queue::create(queue *p) {
      p -> front = p -> rear = -1;
      cout << "\nQueue is created.. " << endl;
}

void   queue::isempty(queue *p) {
```

```
    cout << ((p->front == p->rear)? "\nQueue is Empty..": "\nQueue is Not empty..") <<
endl;
}

void   queue::isfull(queue *p) {
    cout << ((p->rear == max - 1) ? "\nQueue is Full.." : "\nQueue is Not Full..") << endl;
}

void   queue::insert(queue *p, int ele) {
     cout << ((p->rear == max - 1)? "\nQueue Overflows..": (p -> item[++ p->rear] = ele,
"\nElement is Inserted..."));
}

int    queue::remove(queue *p) {
    return (p->front == p->rear)
      ? (cout << "\nQueue Underflows.." << endl, 0)
      : p -> item[++p->front];
}

void   queue::display(queue *p) {
    cout << "\nElements: ";
    for (int i = p->front+1; i <= p->rear; i++)
       cout << p -> item[i] << "  ";
    cout << endl;
}

int    main() {
```

```
    int    ch, ele;
    queue  obj, q;
    queue  *p = &q;

    do {
       cout << "\n1: Create \n2: Is FULL \n3: Is EMPTY \n4: Insert \n5: Remove \n6: Display
\n7: Exit\nEnter your choice: ";
       cin >> ch;

       switch (ch)
       {
          case 1:
             obj.create(p);
             break;

          case 2:
             obj.isfull(p);
             break;

          case 3:
             obj.isempty(p);
             break;

          case 4:
             cout << "\nEnter element to insert: ";
             cin >> ele;
             obj.insert(p, ele);
             break;
```

```
            case 5:
                cout << "\nRemoved Element: " << obj.remove(p)<< endl;
                break;

            case 6:
                obj.display(p);
                break;

            case 7:
                cout << "\nExited" << endl;
                break;

            default:
                cout << "\nWrong Input" << endl;
                break;
        }
    } while (ch != 7);
    return 0;
}
```


Output =>

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 1

Queue is created..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 2

Queue is Not Full..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 3

Queue is Empty..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 100

Element is Inserted..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 200

Element is Inserted..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 300

Element is Inserted..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 6

Elements: 100 200 300

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 2

Queue is Not Full..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 3

Queue is Not empty..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 400

Element is Inserted..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 500

Element is Inserted..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 2

Queue is Full..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 3

Queue is Not empty..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 6

Elements: 100 200 300 400 500

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 5

Removed Element: 100

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 5

Removed Element: 200

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 5

Removed Element: 300

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 6

Elements: 400 500

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 7

Exited

2) Write a program to implement Circular queue.

->

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

class  queue {
     int item[max], front, rear;
     public:
       void   create(queue*);
       void   isempty(queue*);
       void   isfull(queue*);
       void   insert(queue*, int);
       int    remove(queue*);
       void   display(queue*);
};

void   queue::create(queue *p) {
      p -> front = p -> rear = -1;
      cout << "\nQueue is created.. " << endl;
```

```
}

void   queue::isempty(queue *p) {
      cout << ((p->front == p->rear)? "\nQueue is Empty..": "\nQueue is Not empty..") <<
endl;
}

void   queue::isfull(queue *p) {
       cout << (((p->rear == max - 1 && p->front == 0) || (p->front == p->rear + 1))
             ? "\nQueue is Full.."
             : "\nQueue is Not Full..")
        << endl;
}

void   queue::insert(queue *p, int ele) {
       if ((p->front == 0 && p->rear == max - 1) || (p->front == p->rear + 1)) {
          cout << "Queue overflows...\n";
       }
       else {
          if (p->front == -1) p->front = 0;
          p->rear = (p->rear == -1) ? 0 : (p->rear + 1) % max;
          p->item[p->rear] = ele;
          cout << "Element is inserted " << endl;
       }
}

int    queue::remove(queue *p) {
      if (p->front == -1) {
```

```
        cout << "Queue Underflows...\n";
        return 0;
      }
      int z = p->item[p->front];
      p->front = (p->front == p->rear) ? (p->rear = -1, -1) : (p->front + 1) % max;
      return z;
}

void   queue::display(queue *p) {
      cout << "\nElements: ";
      int i = p->front;
       do {
          cout << "\t" << p->item[i];
          i = (i + 1) % max;
       } while (i != (p->rear + 1) % max);
   cout << endl;
}

int    main() {

    int    ch, ele;
    queue  obj, q;
    queue  *p = &q;

    do {
       cout << "\n1: Create \n2: Is FULL \n3: Is EMPTY \n4: Insert \n5: Remove \n6: Display
\n7: Exit\nEnter your choice: ";
       cin >> ch;
```

```
switch (ch)
{
    case 1:
        obj.create(p);
        break;

    case 2:
        obj.isfull(p);
        break;

    case 3:
        obj.isempty(p);
        break;

    case 4:
        cout << "\nEnter element to insert: ";
        cin >> ele;
        obj.insert(p, ele);
        break;

    case 5:
        cout << "\nRemoved Element: " << obj.remove(p)<< endl;
        break;

    case 6:
        obj.display(p);
```

```
            break;

        case 7:
            cout << "\nExited" << endl;
            break;

        default:
            cout << "\nWrong Input" << endl;
            break;
      }
    } while (ch != 7);
    return 0;
}
```

Output =>

```
1: Create
2: Is FULL
3: Is EMPTY
4: Insert
5: Remove
6: Display
7: Exit
Enter your choice: 1

Queue is created..

1: Create
2: Is FULL
```

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 2

Queue is Not Full..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 3

Queue is Empty..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 10

Element is inserted

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 20

Element is inserted

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 30

Element is inserted

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 40

Element is inserted

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 50

Element is inserted

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 2

Queue is Full..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 3

Queue is Not empty..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 60

Queue overflows...

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 6

Elements: 10 20 30 40 50

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 5

Removed Element: 10

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 6

Elements: 20 30 40 50

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 5

Removed Element: 20

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 6

Elements: 30 40 50

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 100

Element is inserted

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 6

Elements: 30 40 50 100

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 200

Element is inserted

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 2

Queue is Full..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 6

Elements: 30 40 50 100 200

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 4

Enter element to insert: 300

Queue overflows...

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove

6: Display

7: Exit

Enter your choice: 7

Exited

3) Write a program to implement IRD (Input Restricted Deque)

->

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

class  Ird {
     int item[max], left, right;
     public:
       void   create(Ird*);
```

```
        void   isempty(lrd*);
        void   isfull(lrd*);
        void   insert(lrd*, int);
        int    remove_left(lrd*);
        int    remove_right(lrd*);
        void   display(lrd*);
};

void   lrd::create(lrd *p) {
      p -> left = p -> right = -1;
      cout << "\nlrd is created.. " << endl;
}

void   lrd::isempty(lrd *p) {
      cout << ((p->left == p->right)? "\nlrd is Empty..": "\nlrd is Not empty..") << endl;
}

void   lrd::isfull(lrd *p) {
      cout << ((p->right == max - 1) ? "\nlrd is Full.." : "\nlrd is Not Full..") << endl;
}

void   lrd::insert(lrd *p, int ele) {
       cout << ((p->right == max - 1)? "\nlrd Overflows..": (p -> item[++ p->right] = ele,
"\nElement is Inserted.."));
}

int    lrd::remove_left(lrd *p) {
      return (p->left == p->right)
```

```
        ? (cout << "\nIrd Underflows.." << endl, 0)
        : p -> item[++p->left];
}

int    Ird::remove_right(Ird *p) {
       return (p->left == p->right)
         ? (cout << "\nIrd Underflows.." << endl, 0)
         : p -> item[p->right--];
}

void   Ird::display(Ird *p) {
       cout << "\nElements: ";
       for (int i = p->left+1; i <= p->right; i++)
          cout << p -> item[i] << "  ";
       cout << endl;
}

int    main() {

    int    ch, ele;
    Ird  obj, q;
    Ird  *p = &q;

    do {
       cout << "\n1: Create \n2: Is FULL \n3: Is EMPTY \n4: Insert \n5: Remove_left \n6:
Remove_right \n7: Display \n8: Exit\nEnter your choice: ";
       cin >> ch;
```

```
switch (ch)
{
    case 1:
        obj.create(p);
        break;

    case 2:
        obj.isfull(p);
        break;

    case 3:
        obj.isempty(p);
        break;

    case 4:
        cout << "\nEnter element to insert: ";
        cin >> ele;
        obj.insert(p, ele);
        break;

    case 5:
        cout << "\nRemoved Element: " << obj.remove_left(p)<< endl;
        break;

    case 6:
        cout << "\nRemoved Element: " << obj.remove_right(p)<< endl;
        break;
```

```
            case 7:
                obj.display(p);
                break;

            case 8:
                cout << "\nExited" << endl;
                break;

            default:
                cout << "\nWrong Input" << endl;
                break;
        }
    } while (ch != 8);
    return 0;
}
```


Output =>

```
1: Create
2: Is FULL
3: Is EMPTY
4: Insert
5: Remove_left
6: Remove_right
7: Display
8: Exit
Enter your choice: 1
```

Ird is created..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 2

Ird is Not Full..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 3

Ird is Empty..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 4

Enter element to insert: 100

Element is Inserted..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 4

Enter element to insert: 200

Element is Inserted..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 4

Enter element to insert: 300

Element is Inserted..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 4

Enter element to insert: 400

Element is Inserted..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 7

Elements: 100 200 300 400

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 5

Removed Element: 100

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 7

Elements: 200 300 400

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 6

Removed Element: 400

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 7

Elements: 200 300

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 5

Removed Element: 200

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 7

Elements: 300

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 6

Removed Element: 300

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 7

Elements:

1: Create

2: Is FULL

3: Is EMPTY

4: Insert

5: Remove_left

6: Remove_right

7: Display

8: Exit

Enter your choice: 8

Exited

4) Write a program to implement ORD (Output Restricted Deque)

->

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

class  Ord {
     int item[max], left, right;
     public:
       void   create(Ord*);
       void   isempty(Ord*);
       void   isfull(Ord*);
       void   insert_left(Ord*, int);
       void   insert_right(Ord*, int);
       int    remove(Ord*);
       void   display(Ord*);
};

void   Ord::create(Ord *p) {
      p -> left = p -> right = -1;
      cout << "\nOrd is created.. " << endl;
}

void   Ord::isempty(Ord *p) {
      cout << ((p->left == p->right)? "\nOrd is Empty..": "\nOrd is Not empty..") << endl;
}
```

```
void   Ord::isfull(Ord *p) {
      cout << ((p->right == max - 1) ? "\nOrd is Full.." : "\nOrd is Not Full..") << endl;
}

void   Ord::insert_left(Ord *p, int ele) {
      if (p->right == max - 1) {
        cout << "\nOrd Overflows.." << endl;
      }
      else {
        for (int i = p->right + 1; i >= p->left + 2; i --)
            p->item[i] = p->item[i-1];
        p->item[p->left+1] = ele;
        p->right++;
        cout << "\nElement is Inserted form left.." << endl;
      }
}

void   Ord::insert_right(Ord *p, int ele) {
       cout << ((p->right == max - 1)? "\nOrd Overflows..": (p -> item[++ p->right] = ele,
"\nElement is Inserted form right..")) << endl;
}

int    Ord::remove(Ord *p) {
      return (p->left == p->right)
        ? (cout << "\nOrd Underflows.." << endl, 0)
        : p -> item[++p->left];
}
```

```
void   Ord::display(Ord *p) {
       cout << "\nElements: ";
       for (int i = p->left+1; i <= p->right; i++)
          cout << p -> item[i] << "  ";
       cout << endl;
}

int    main() {

    int    ch, ele;
    Ord  obj, q;
    Ord  *p = &q;

    do {
       cout << "\n1: Create \n2: Is FULL \n3: Is EMPTY \n4: Insert_left \n5: Insert_right \n6:
Remove \n7: Display \n8: Exit\nEnter your choice: ";
       cin >> ch;

       switch (ch)
       {
          case 1:
             obj.create(p);
             break;

          case 2:
             obj.isfull(p);
             break;
```

```
case 3:
    obj.isempty(p);
    break;

case 4:
    cout << "\nEnter element to insert: ";
    cin >> ele;
    obj.insert_left(p, ele);
    break;

case 5:
    cout << "\nEnter element to insert: ";
    cin >> ele;
    obj.insert_right(p, ele);
    break;

case 6:
    cout << "\nRemoved Element: " << obj.remove(p)<< endl;
    break;

case 7:
    obj.display(p);
    break;

case 8:
    cout << "\nExited" << endl;
    break;
```

```
            default:
                cout << "\nWrong Input" << endl;
                break;
        }
    } while (ch != 8);
    return 0;
}
```

Output =>

```
1: Create
2: Is FULL
3: Is EMPTY
4: Insert_left
5: Insert_right
6: Remove
7: Display
8: Exit
Enter your choice: 1

Ord is created..

1: Create
2: Is FULL
3: Is EMPTY
4: Insert_left
5: Insert_right
```

6: Remove

7: Display

8: Exit

Enter your choice: 2

Ord is Not Full..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 3

Ord is Empty..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 4

Enter element to insert: 100

Element is Inserted form left..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 4

Enter element to insert: 200

Element is Inserted form left..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 7

Elements: 200 100

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 5

Enter element to insert: 400

Element is Inserted form right..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 5

Enter element to insert: 600

Element is Inserted form right..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 7

Elements: 200 100 400 600

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 4

Enter element to insert: 700

Element is Inserted form left..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 7

Elements: 700 200 100 400 600

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 4

Enter element to insert: 900

Ord Overflows..

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 7

Elements: 700 200 100 400 600

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 6

Removed Element: 700

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 7

Elements: 200 100 400 600

1: Create

2: Is FULL

3: Is EMPTY

4: Insert_left

5: Insert_right

6: Remove

7: Display

8: Exit

Enter your choice: 8

Exited

5) Write a program to implement Priority queue.

->

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

class   queue {
    int  item[max], pri[max], front, rear;
    public:
      void    create(queue*);
      void    insert(queue*, int, int);
      int     remove(queue*);
      void    display(queue*);
};
```

```
void   queue::create(queue *p) {
       p -> front = p -> rear = -1;
       cout << "\nQueue is created.." << endl;
}

void   queue::insert(queue *p, int ele, int pri) {
       if (p->rear == max - 1)
          cout << "\nQueue Overflows.." << endl;
       else {
           ++p->rear;
           p->item[p->rear] = ele;
           p->pri[p->rear] = pri;
           cout << "\nElement is inserted.." << endl;
       }
}

int    queue::remove(queue *p) {
       int m = p->pri[p->front+1], pos = 0;
       if (p->front == p->rear) {
          cout << "\nQueue Underflows..";
          return 0;
       }
       else {

          for (int i = p->front+1; i <= p->rear; i++) {
              if (m < p->pri[i]) {
```

```
            m = p->pri[i];
            pos = i;
          }
        }
        int ele = p->item[pos];
        for (int i = pos; i <= p->rear; i++){
          p->item[i] = p->item[i+1];
          p->pri[i] = p->pri[i+1];
        }
        --p->rear;
        return ele;
      }
}

void   queue::display(queue *p) {
      cout << "\nElements      Priority" << endl;
      for (int i = p->front+1; i <= p->rear; i++) {
        cout << p->item[i] << "           " << p->pri[i] << endl;
      }
      cout << endl;
}

int    main() {

    int    ch, ele, pri;
    queue  obj, q;
    queue  *p = &q;
```

```
    do {
        cout << "\n1: Create \n2: Insert \n3: Remove \n4: Display \n5: Exit\nEnter your
choice: ";
        cin >> ch;

        switch (ch)
        {
            case 1:
                obj.create(p);
                break;

            case 2:
                cout << "\nEnter element: ";
                cin >> ele;
                cout << "Enter Priority: ";
                cin >> pri;
                obj.insert(p, ele, pri);
                break;

            case 3:
                cout << "\nRemoved Element: " << obj.remove(p)<< endl;
                break;

            case 4:
                obj.display(p);
                break;
```

```
            case 5:
                cout << "\nExited.." << endl;
                break;

            default:
                cout << "\nWrong Input" << endl;
                break;
        }
    } while (ch != 5);
    return 0;
}
```

Output =>


```
1: Create
2: Insert
3: Remove
4: Display
5: Exit
Enter your choice: 1

Queue is created..

1: Create
2: Insert
3: Remove
4: Display
5: Exit
Enter your choice: 2
```

Enter element: 20

Enter Priority: 4

Element is inserted..

1: Create

2: Insert

3: Remove

4: Display

5: Exit

Enter your choice: 2

Enter element: 10

Enter Priority: 1

Element is inserted..

1: Create

2: Insert

3: Remove

4: Display

5: Exit

Enter your choice: 2

Enter element: 30

Enter Priority: 2

Element is inserted..

1: Create

2: Insert

3: Remove

4: Display

5: Exit

Enter your choice: 4

Elements Priority

20 4

10 1

30 2

1: Create

2: Insert

3: Remove

4: Display

5: Exit

Enter your choice: 3

Removed Element: 20

1: Create

2: Insert

3: Remove

4: Display

5: Exit

Enter your choice: 4

Elements Priority

10 1

30 2

1: Create

2: Insert

3: Remove

4: Display

5: Exit

Enter your choice: 3

Removed Element: 30

1: Create

2: Insert

3: Remove

4: Display

5: Exit

Enter your choice: 4

Elements Priority

10 1

1: Create

2: Insert

3: Remove

4: Display

5: Exit

Enter your choice: 3

Removed Element: 10

1: Create

2: Insert

3: Remove

4: Display

5: Exit

Enter your choice: 4

Elements Priority

1: Create

2: Insert

3: Remove

4: Display

5: Exit

Enter your choice: 5

Exited..