```
#include <bits/stdc++.h>
using namespace std;
const int MAX = 100;
#define MAX_SIZE 1000
int Queue[MAX_SIZE];
int i=0;

class PQueue{
    private:
        int PQ_data[MAX];
        int PQ_priority[MAX];
        int front = -1, rear = -1;
    public:
        void _witharray_add();
        void _witharray_remove();
        void _witharray_print();
};

class Node {
    public:
        int data;
        int priority;
        Node *next;
};
// Function to create a new node
Node* newNode(int d, int p)
{
    Node* temp = (Node*)malloc(sizeof(Node));
    temp->data = d;
    temp->priority = p;
    temp->next = NULL;
    return temp;
}

// Returnning the head value
int peek(Node** head)
{
    return (*head)->data;
}

// Removes the element with the
// highest priority form the list
void pop(Node** head)
{
    Node* temp = *head;
    (*head) = (*head)->next;
    free(temp);
}

// Function to push according to priority
void push(Node** head, int d, int p)
{
    Node* start = (*head);
    // Create new Node
    Node* temp = newNode(d, p);
    if ((*head)->priority > p)
    {
        // Insert New Node before head
        temp->next = *head;
        (*head) = temp;
    }
    else
    {

        // Traverse the list and find a

        // position to insert new node
        while (start->next != NULL &&
               start->next->priority < p)
        {
```

```
68          start = start->next;
69      }
70
71      // Either at the ends of the list
72      // or at required position
73      temp->next = start->next;
74      start->next = temp;
75  }
76  }
77
78  // Function to check is list is empty
79  int isEmpty(Node** head)
80  {
81      return (*head) == NULL;
82  }
83
84  void PQueue::_witharray_add(){
85      int element;
86      int Pr=0;
87      int i=0;
88      if(front==0 && rear== MAX-1){
89          cout << "\n Case of Overflow!!!";
90      }
91      else{
92      cout << "Enter the data and it's priority to be added: -";
93      cin >> element>> Pr;
94      // ie. if this is the first element to be addeds into the queue
95          if(front == -1){
96              front = rear =0;
97              PQ_data[rear] = element;
98              PQ_priority[rear] = Pr;
99          }
100         // some elememts are in the list
101         else if(rear == MAX-1){
102             for(i=front;i<=rear;i++){
103                 // shifting the element to the next index
104                 PQ_data[i-front] = PQ_data[i];
105                 PQ_priority[i-front] = PQ_priority[i];
106                 rear = rear-front;
107                 for(i = rear;i>front;i--){
108                     if(Pr>PQ_priority[i]){
109                         PQ_data[i+1] = PQ_data[i];
110                         PQ_priority[i+1] = PQ_priority[i];
111                     }
112                     else
113                         break;
114
115                     PQ_data[i+1]=element;
116                     PQ_priority[i+1]=Pr;
117                     rear++;
118                 }
119             }
120         }
121
122         else{
123             for(i=rear;i>=front;i--){
124                 if(Pr>PQ_priority[i]){
125                     // the input priortity is grt than the preset
126                     // shift the at i to i+1
127                     PQ_data[i+1]=PQ_data[i];
128                     PQ_priority[i+1] = PQ_priority[i];
129                 }
130                 else
131                     break;
132                 }
133                 // if the input priority is not grt than add that element into the next index
134                 PQ_data[i+1]=element;
135                 PQ_priority[i+1] = Pr;
136                 rear++;
137
138
```

```
        }
    }
}

void PQueue::_witharray_print(){
    int i=0;
    for(int i=front;i<=rear;i++){
        cout << "Element = "<< PQ_data[i] << "\t "<< "Priority = "<< PQ_priority[i]<< "\n";
    }
}

void PQueue::_witharray_remove() //remove the data from front
{
 if(front == -1)
 {
 cout<<"Queue is Empty";
 }
 else
 {
 cout<<"Deleted Element ="<<PQ_data[front]<<endl;
 cout<<"Its Priority = "<<PQ_priority[front]<<endl;
  if(front==rear)
   front = rear = -1;
  else
   front++;
 }
}

void Enqueue_using_array(int item) {
        // Check if the queue is full
        if (i==MAX_SIZE-1) {
             cout<<"Error:Queue is full\n";
              return;
        }
        Queue[i++] = item;
}
/* Removes the item with the maximum priority
 search the maximum item in the array and replace it with
the last item ,In worst case time complexity approaches
to o(n).*/

int Dequeue_using_array()
{
    int item;
        // Check if the queue is empty
        if (i == 0) {
              cout<<"ERROR:Queue is empty\n";
              return -999999;
        }
        int j, maxi = 0;
        // find the maximum priority
        for (j = 1; j < i; j++) {
              if (Queue[maxi] < Queue[j]) {
                    maxi = j;
              }
        }
        item = Queue[maxi];


        // replace the max with the last element
        Queue[maxi] = Queue[i - 1];
        i = i - 1;
        return item;


}
void Display_using_array(int Queue[],int i)
{
  cout<<"The priority queue at this stage is as follows:-\n";
    for(int j=0;j<i;j++)
    {
```

```
        cout<<Queue[j]<<" ";
    }
    cout<<"\n";
}

void  PriorityQueuebyArray(int Queue[])
{   int count;
     while(true)
     {
        int option;
        cout<<"1. Enter 1 for insert operation\n";
        cout<<"2. Enter 2 for remove operation\n";
        cout<<"3. Enter 3 for displaying of priority queue\n";
        cout<<"4. To Quit\n";
        cin>>option;
        if(option==1)
        {   int x;
            cout<<"Enter the vaalue to insert:";
            cin>>x;
            Enqueue_using_array(x);
            count--;
        }
        else if(option==2)
        {
           cout<<"Deleted element is"<<Dequeue_using_array()<<"\n";
           count--;
        }
        else if(option==3)
        {
            Display_using_array(Queue,i);
            count--;
        }
        else{
          cout<<"Thansk a lot!! Quitting....";
          break;
        }
     }
}

int main(){
    Node* pq = newNode(4,1);
    int choice;
    cout << "Enter\n 1. Implementation with Multiple Arrays\n 2. Implementation with List\n 3. Implementation with Single Array \n";
    cin >> choice;
    if(choice ==1){
        PQueue *__withArray = new PQueue();
        while(true){
            cout << "Enter \n 1. To add an element \n 2. Remove the element \n 3. To display 4. Quit \n";
            cin >> choice;
            if(choice ==1){
                __withArray->_witharray_add();
            }
            else if(choice ==2){
                __withArray->_witharray_remove();
            }
            else if(choice ==3){
                __withArray->_witharray_print();
            }
            else{
                break;
            }
        }
    }
    else if(choice ==2){
       while(true){
            cout << "Enter \n 1. To add an element \n 2. Remove the element \n 3. To display \n  4. Quit \n";
            cin >> choice;
            int data=0;
            int prio=0;
            // created a default node with value -1
```

```
280         if(choice ==1){
281             cout << "Enter the value of Elemnet and it's priority :- ";
282             cin >>data >> prio;
283             push(&pq,data,prio);
284         }
285         else if(choice ==2){
286            cout<< "Popping out the element!!" << peek(&pq);
287         }
288         else if(choice ==3){
289            while (!isEmpty(&pq)) {
290            cout << " " << peek(&pq)<< "";
291            pop(&pq);
292            }
293            cout << "\n";
294         }
295         else{
296            break;
297         }
298       }
299    }
300    else if(choice ==3){
301       PriorityQueuebyArray(Queue);
302    }
303    else{
304       cout << "Enter a valid input \n Quiting...";
305    }
}
```