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

template <class T> class Queue {

    template <class U> struct Node {

        int priority;
        U value;
        Node<U> *next;
        Node(int priority, U value)
        :priority (priority), value(value) ,next(nullptr) { }
};
    Node<T> *head;

public:
    Queue()
    :head (nullptr) { } void enqueue(int priority, T value)
    {

        Node<T> *newNode = new Node<T>(priority, value);

        if(nullptr == head) {
            head = newNode;
            printQueue();
            return;
        }

        Node<T> *back = head;

        for(Node<T> *curr = head; curr != nullptr; curr = curr->next)
        {
        if(priority >= curr->priority)
        {
            back = curr;
            if(nullptr == curr->next)
            {
                back->next = newNode;
                printQueue();
                return;
            }
        }
        else
        {
        if(curr == head)
        {
            newNode->next = curr;
            head = newNode;
            printQueue();
            return;
        }

        else
        {
            newNode->next = curr;
            back->next = newNode;
            printQueue();
            return;
        }
        }
}
}
    bool dequeue()
    {
        Node<T> *curr = head;
        head = head->next;
```

```
        cout << curr->value << endl;
        delete curr;
        return true;

    }

    void printQueue()
    {
        Node<T> *curr = head;
        while(curr)
        {
                cout << curr->value << "\t";
                curr = curr->next;
        }
        cout << endl;
    }
};

int main() {
    int value;
    int priority;
    Queue<int> q1;
    while(cout << "Enter value in queue (0 to stop) : ",
    cin >> value,
    value)
    {
        cout << "Enter priority : ";
        cin >> priority;
        q1.enqueue(priority, value);

    }
    q1.dequeue();
    return 0;
}
```