

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 #include <cstring>
5 using namespace std;
6 template <class t1>
7 struct node
8 {
9     t1 info;
10    node* next;
11 };
12
13 template <class t1>
14 class queues
15 {
16     node<t1>* front;
17     node<t1>* rear;
18 public:
19     queues();
20     void enqueue(t1);
21     t1 delqueue();
22     bool isfull();
23     bool isempty();
24     void print(ofstream&);
25 };
26
27 template<class t1>
28 queues<t1>::queues()
29 {
30     front = NULL;
31     rear = NULL;
32 }
33
34 template<class t1>
35 void queues<t1>::enqueue(t1 a)
36 {
37     node<t1>* temp;
38     temp = new node<t1>;
39     temp->info = a;
40     temp->next = NULL;
41     if (rear == NULL)
42         front = temp;
43     else
44         rear->next = temp;
45
46     rear = temp;
47 }
48
49 template<class t1>
```

```
50 t1 queues<t1>::delqueue()
51 {
52     t1 value;
53     node<t1>* temp;
54     value = front->info;
55     temp = front;
56     front = front->next;
57     temp->next = NULL;
58     delete temp;
59     return value;
60 }
61
62 template<class t1>
63 bool queues<t1>::isfull()
64 {
65     node<t1>* ptr;
66     ptr = new node<t1>;
67     if (ptr == NULL)
68     {
69         return true;
70     }
71     else
72     {
73         delete ptr;
74         return false;
75     }
76 }
77
78 template<class t1>
79 bool queues<t1>::isempty()
80 {
81     if (front == NULL)
82         return true;
83     else
84         return false;
85 }
86
87 template<class t1>
88 void queues<t1>::print(ofstream& fout)
89 {
90     node<t1>* temp;
91     temp = front;
92     while (temp != NULL)
93     {
94         fout << temp->info;
95         temp = temp->next;
96     }
97 }
98
```