```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
using namespace std;
template <class t1>
struct node
{
    t1 info;
    node* next;
};

template <class t1>
class queues
{
    node<t1>* front;
    node<t1>* rear;
public:
    queues();
    void enqueue(t1);
    t1 delqueue();
    bool isfull();
    bool isempty();
    void print();
};

template<class t1>
queues<t1>::queues()
{
    front = NULL;
    rear = NULL;
}

template<class t1>
void queues<t1>::enqueue(t1 a)
{
    node<t1>* temp;
    temp = new node<t1>;
    temp->info = a;
    temp->next = NULL;
    if (rear == NULL)
        front = temp;
    else
    {
        rear->next = temp;
        rear = temp;
    }
}

```

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