```
struct QueueException {                                  //Queue Exception
    QueueException(char* m)
    {   cout << endl;
        cout << "I am the Queue and I am " << m << ".";
        cout << endl;
    }
};
template <class T>
class Queue {
    int size;                                            //Number of available elements
    int count;                                           //Number of occupied elements
    int oldest;                                          //Index of the oldest element
    int newest;                                          //Index of the newest element
    T* Q;                                                //Points to storage for the Queue
public:
    //-------------------------------------------------------------------------------------------------------------
    //Constructor, allocate storage
    //-------------------------------------------------------------------------------------------------------------
    Queue(int sz=100):size(sz),count(0),oldest(0),newest(-1){Q=new T[size];}
    ~Queue(){if (Q) delete[] Q;}                         //Destructor, reclaim storage
    bool IsFull(void){return count>=size;}               //Is the Queue full?
    bool IsEmpty(void){return count<=0;}                 //Is the Queue empty?
    void Enq(T v)                                        //Put value v at the newest end of the Q
    {   if (IsFull()) throw QueueException("full");
        newest=(newest+1)%size;
        Q[newest]=v;
        count++;
    }
    T Deq(void)                                          //Remove and return the oldest value
    {   if (IsEmpty()) throw QueueException("empty");    //on the Queue
        T v=Q[oldest];
        oldest=(oldest+1)%size;
        count--;
        return v;
    }
};
```

Figure 1. template class *Queue*.

Example: Program **p01** stores and retrieves the characters of string "Alan Turing" illustrating the first-in-first-out character of a queue.

```
#ifndef Queue_h
#define Queue_h 1
struct QueueException {                                  //Queue Exception
    QueueException(char* m)
    {   cout << endl;
        cout << "I am the Queue and I am " << m << ".";
        cout << endl;
    }
};
template <class T>
class Queue {
    int size;                                            //Number of available elements
    int count;                                           //Number of occupied elements
    int oldest;                                          //Index of the oldest element
    int newest;                                          //Index of the newest element
    T* Q;                                                //Points to storage for the Queue
public:
    //-------------------------------------------------------------------------------------------------------------
    //Constructor, allocate storage
    //-------------------------------------------------------------------------------------------------------------
    Queue(int sz=100):size(sz),count(0),oldest(0),newest(-1){Q=new T[size];}
    ~Queue(){if (Q) delete[] Q;}                         //Destructor, reclaim storage
    bool IsFull(void){return count>=size;}               //Is the Queue full?
    bool IsEmpty(void){return count<=0;}                 //Is the Queue empty?
    void Enq(T v)                                        //Put value v at the newest end of the Q
    {   if (IsFull()) throw QueueException("full");
        newest=(newest+1)%size;
        Q[newest]=v;
        count++;
    }
    T Deq(void)                                          //Remove and return the oldest value
    {   if (IsEmpty()) throw QueueException("empty");    //on the Queue
        T v=Q[oldest];
        oldest=(oldest+1)%size;
        count--;
        return v;
    }
};
#endif
```

Figure 2. File **Queue.h**

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
#include "Queue.h"
int main()
{     string t="Alan Turing";
      Queue <char> Q;
      for (int a=0;a<t.length();a++) Q.Enq(t[a]);
      cout << endl;
      while (!Q.IsEmpty()) cout << Q.Deq();
      cout << endl;
      return 0;
}
```

Figure 3. File **p01.cpp**

Program **p01** prints:
Alan Turing