```
 1  /* Filename: sutr.cpp
 2   * ------------------
 3   * Author:  Darwin Jacob Groskleg
 4   * Class:   CSCI 162
 5   * Lab:     #19
 6   * Date:    Friday, March 9th, 2018
 7   *
 8   * Purpose: to implement an example of a queue data structure being used to
 9   * solve a problem. Following the given algorithm and using an intQueue
10   * determine the suiter that gets to marry the princess. Do this for suiter
11   * line sizes of 2 through 20.
12   *
13   * Console
14   * ------------------------------------------------------------------------
15   * $> ./sutr
16   * Out of a line of 2 suitors, princess Eve selects suitor 2
17   * Out of a line of 3 suitors, princess Eve selects suitor 2
18   * Out of a line of 4 suitors, princess Eve selects suitor 1
19   * Out of a line of 5 suitors, princess Eve selects suitor 4
20   * Out of a line of 6 suitors, princess Eve selects suitor 1
21   * Out of a line of 7 suitors, princess Eve selects suitor 4
22   * Out of a line of 8 suitors, princess Eve selects suitor 7
23   * Out of a line of 9 suitors, princess Eve selects suitor 1
24   * Out of a line of 10 suitors, princess Eve selects suitor 4
25   * Out of a line of 11 suitors, princess Eve selects suitor 7
26   * Out of a line of 12 suitors, princess Eve selects suitor 10
27   * Out of a line of 13 suitors, princess Eve selects suitor 13
28   * Out of a line of 14 suitors, princess Eve selects suitor 2
29   * Out of a line of 15 suitors, princess Eve selects suitor 5
30   * Out of a line of 16 suitors, princess Eve selects suitor 8
31   * Out of a line of 17 suitors, princess Eve selects suitor 11
32   * Out of a line of 18 suitors, princess Eve selects suitor 14
33   * Out of a line of 19 suitors, princess Eve selects suitor 17
34   * Out of a line of 20 suitors, princess Eve selects suitor 20
35   * ------------------------------------------------------------------------
36   */
37
38  #include <iostream>
39  #include <iomanip>
40
41  #include "intQueue.h"
42
43  using namespace std;
44
45  int EvesChoiceFrom(int);
46  void FormulaAnalysis();
47
48  int main() {
49      for (int suitors=2; suitors<=20; suitors++) {
50          cout << "Out of a line of " << suitors
51               << " suitors, princess Eve selects suitor "
52               << EvesChoiceFrom(suitors)
53               << endl;
54      }
55
56      return 0;
57  }
58
59
60  /* Function Name: EvesChoiceFrom
61   * Usage: int choice = EvesChoiceFrom(7);
62   * -------------------------------------
63   * Implements the algorithm for how the princess Eve selects a mate out of a
64   * line of suitors.
65   *
66   *  1. Move first 2 suitors to end of line
67   *  2. Eliminate 3rd suitor
68   *  3. Repeat until 1 suitor remains.
69   *
```

```
 70   * General Formula:
 71   *
 72   *
 73   *
 74   *
 75   *
 76   *
 77   *
 78   *
 79   */
 80  int EvesChoiceFrom(int line_size) {
 81      const int SKIPS = 2;
 82      int skips_left = SKIPS;
 83      int suitor;
 84      intQueue suitor_line(line_size);
 85
 86      for (int s=1; s<=line_size; s++)
 87          suitor_line.Enqueue(s);
 88
 89      while (suitor_line.Size() > 1) {
 90          suitor = suitor_line.Front();
 91          suitor_line.Dequeue();
 92          if (skips_left > 0) {
 93              suitor_line.Enqueue(suitor);
 94              skips_left--;
 95          }
 96          else
 97              skips_left = SKIPS;
 98      }
 99      return suitor_line.Front();
100  }
101
102
103  /* FunctionName: formulaAnalysis
104   * ----------------------------
105   * NOT CALLED
106   * Implemented as a spike for attemting to find a pattern in the suitor
107   * selection formula.
108   * See EvesChoiceFrom for general formula.
109   */
110  void FormulaAnalysis() {
111      int cycle = 0;
112      int cycle_size = 0;
113      int last_choice = 100;
114
115      for (int suitors=1; suitors<=100; suitors++) {
116          int choice = EvesChoiceFrom(suitors);
117          if (choice < last_choice) {
118              cycle++;
119              cycle_size = 1;
120              cout << " ---- ---- --- ---\n";
121          } else {
122              cycle_size++;
123          }
124          last_choice = choice;
125
126          cout << setw(5) << suitors
127               << setw(5) << choice
128               << setw(4) << cycle
129               << setw(4) << cycle_size
130               << endl;
131      }
132  }
```

```
 1  /* Filename: intQueue.h
 2   * --------------------
 3   * Author:  Darwin Jacob Groskleg
 4   * Class:   CSCI 162
 5   * Lab:     #19
 6   * Date:    Friday, March 9th, 2018
 7   *
 8   * Purpose: interface for intQueue objects, queue data structures for integers
 9   */
10
11  #ifndef INT_Queue_H_INCLUDED
12  #define INT_Queue_H_INCLUDED
13
14  class intQueue {
15      private:
16          int currentsize;
17          int arraysize;
18          int first;
19          int last;
20          int *queuearray;
21      public:
22          intQueue(int maxsize = 100);
23          ~intQueue() { delete [] queuearray; }
24
25          bool Enqueue(int);
26          bool Dequeue();
27
28          int  Front() const;
29
30          int  Size()  const { return currentsize; }
31
32          bool Empty() const { return currentsize == 0; }
33          bool Full()  const { return currentsize == arraysize; }
34  };
35
36  #endif // INT_Queue_H_INCLUDED
```

```
 1  /* Filename: intQueue.cpp
 2   * ----------------------
 3   * Author:  Darwin Jacob Groskleg
 4   * Class:   CSCI 162
 5   * Lab:     #19
 6   * Date:    Friday, March 9th, 2018
 7   *
 8   * Purpose: implements the methods on intQueue class.
 9   */
10
11  #include "intQueue.h"
12
13  intQueue::intQueue(int maxsize) {
14      queuearray  = new int[maxsize];
15      currentsize = first = last = 0;
16      arraysize   = maxsize;
17  }
18
19  bool intQueue::Enqueue(int value) {
20      if (Full()) return false;
21
22      queuearray[last] = value;
23      last = (last + 1) % arraysize;
24      currentsize++;
25
26      return true;
27  }
28
29  bool intQueue::Dequeue() {
30      if (Empty()) return false;
31
32      first = (first  + 1) % arraysize;
33      currentsize--;
34
35      return true;
36  }
37
38  int intQueue::Front() const {
39      return queuearray[first];
40  }
```