```
1  // Demonstrate use of C++ Exceptions
2  // George F. Riley, Georgia Tech, Fall 2009
3
4  #include <iostream>
5  #include <vector>
6  #include <string>
7  #include <exception>  // Defines the exception class
8
9  using namespace std;
10
11 class A
12 {
13 public:
14   A(int i1)
15       : i(i1)
16   { // Just for illustration, we throw an exception with char* argument
17     // Since the exception occurs in the constructor, the resulting
18     // object is mal-formed and cannot be accessed legally.
19     throw "this is a test";
20   }
21   int i;
22 };
23
24 // LargeMemoryClass is used to quickly exhaust all available memory
25 class LargeMemoryClass
26 {
27 public:
28   LargeMemoryClass()
29     { // Nothing needed for constructor
30     }
31
32 private:
33   double bigArray[0x100000]; // Takes lots of memory, used to illustrate
34   // out of memory exception
35 };
36
37 // Class MyException shows creating your own exceptions
38 class MyException: public exception
39 {
40   virtual const char* what() const throw()
41   {
42     return "My exception happened";
43   }
44 };
45
46 //void MySub1(int k) throw(int, MyException, string)
47 void MySub1(int k)
48 {
49   if (k <= 100)  throw MyException();
50   if (k <= 1000) throw k;
51   // throw exception not expected by caller
52   throw (string("Hello"));
53 }
54
55 int main()
56 {
```

Program exceptions.cc

```
57    A* a;
58    try
59      {
60        a = new A(1);
61      }
62    catch (const char* p)
63      {
64        //cout << "Exception " << p << " a->i " << a->i << endl; // segfaults
65        cout << "Exception " << p << " a does not exist" << endl;
66      }
67
68
69    // Allocate many LargeMemoryClass objects until memory runs out
70    vector<LargeMemoryClass*> largeMem; // Used to give the memory back
71    LargeMemoryClass* lmc;
72    while(true)
73      {
74        try
75          {
76            lmc = new LargeMemoryClass;
77            largeMem.push_back(lmc);
78          }
79        catch (exception& e)
80          {
81            cout << "exception " << e.what() << endl;
82            // Give back the memory
83            for (unsigned i = 0; i < largeMem.size(); ++i)
84              {
85                delete largeMem[i];
86              }
87            break;
88          }
89      }
90
91    // Illustrate MyException
92    try
93    {
94      throw MyException();
95    }
96    catch (exception& e)
97    {
98      cout << e.what() << endl;
99    }
100
101   // Illustrate calling a subroutine that throws an exception
102   try
103     {
104       //MySub1(100);
105       //MySub1(1000);
106       MySub1(10000);
107     }
108   catch (exception& e)
109     {
110       cout << "MySub1 exception " << e.what() << endl;
111     }
112   catch (int i)
```

Program exceptions.cc (continued)

```
113       {
114         cout << "MySub1 int exception " << i << endl;
115       }
116     catch (...)
117       {
118         cout << "MySub1, unknown exception" << endl;
119       }
120   }
121
122
123
```

Program exceptions.cc (continued)