Com S 327
Fall 2017
Final Exam

DO NOT OPEN THIS EXAM UNTIL INSTRUCTED TO DO SO

Name: _____

ISU NetID (username): _____

Closed book and notes, no electronic devices, no headphones. Time limit 105 minutes. Partial credit may be given for partially correct solutions.

- Use correct C++ syntax for writing code.
- You are not required to write comments for your code; however, brief comments may help make your intention clear in case your code is incorrect.

If you have questions, please ask!

Question	Points	Your Score
1	40	
2	40	
3	20	
EC	3	
Total	100	

1. (40 pts; 5 ea) Give the output of the following code snippets, if any. If the code does not produce output, write *no output*. If the code produces a runtime error, write *error*. None of this code produces compile-time errors. All parts of this problem are independent, except where stated below.

   (a)
```
cout << "Everything is not what it seems" << endl;
```

   (b)
```
string speech = "Later, losers.";
cout << "Justin, are you sure you don't want to use my graduation "
     << "speech?  It goes line this, \"" << speech << "\"\n";
```

   (c)
```
string *s;

s = (string *) "I'm not lazy.  I'm just resting up for my 30s.";
cout << *s << endl;
```

The next two problems use the function `name()`, defined as follows:

```
const char *&name() {
  static const char *n = "Justin";

  cout << n << ".\n";

  return n;
}
```

(d) `name() = "Alex"`

(e) `cout << (name() = "Max") << endl;`

The remaining problems depend on the class hierarchy defined below:

```
class human {
public:
  virtual void print()
  {
    cout << "I am a human." << endl;
  }
  virtual ~human() {}
};

class wizard : public human {
public:
  virtual void print()
  {
    human::print();
    cout << "I am also a wizard." << endl;
  }
};
```

(f)
```
human harper;
harper.print();
```

(g)
```
vector<human> v;
v.push_back(human());
v.push_back(wizard());
v[1].print();
```

(h)
```
vector<human *> v;
v.push_back(new human());
v.push_back(new wizard());
v[1]->print();
```

2. (40 pts) Below is a simple, templated circular queue class. You must implement two new methods according to the specified functionality. Because this is a templated class, your methods are implemented inside the class definition itself. Except for the addition of the two specified methods, you may not otherwise alter the class definition.

```
#include <iostream>
#include <vector>

using namespace std;

template <class T>
class circular_queue {
public:
  vector<T> v;
  int size;
  int front, back;

  int enqueue(T d) {
    if (front != back) {
      v[front++] = d;
      if (front == size) {
        front = 0;
      }
      return 0;
    }
    throw "No space left in queue";
  };

  T dequeue() {
    if (((back + 1) % size) != front) {
      back++;
      if (back == size) {
        back = 0;
      }
    } else {
      throw "Nothing to dequeue";
    }

    return v[back];
  }

  circular_queue(int size) : v(size + 1), size(size + 1),
                             front(0),    back(size) {
  }

  ~circular_queue() {
  }

  // The code you will implement on the following pages goes here!
};
```

(a) (15 pts) Implement the overloaded assignment operator to assign a `circular_queue` to a `circular_queue`. Remember that, given objects `p` and `q`:

```
(p = q).method();
```

is equivalent to:

```
p = q;
p.method();
```

which has implications on the return type of the operator (if you haven't memorized this—which I wouldn't expect you to—then this should be a useful hint).

(b) (25 pts) Implement the method, `int grow()`, which doubles the storage capacity of the queue. Note that simply doubling the size of the vector, in addition to resulting in the wrong size, will leave data in the wrong place(s). A correct solution will have to go into a bit more depth than that.

The method to resize a vector is `resize(int size)`, which, when growing the vector, will preserve the data in the original. The most obvious solution will use `resize()`, but the simplest (not significantly simpler, so don't waste a lot of time thinking about it) will not.

3. (20 pts; 2 ea) Circle TRUE or FALSE in response to each of these statements about C++. Assume that the necessary headers are included for any function or class used. Read every word carefully; some of these are subtle.

   (a) The following line is a valid statement in C++:

   ```
   int *i = (int *) malloc(12 * sizeof (*i));
   ```

   TRUE FALSE

   (b) Some C is not valid C++, but most is.

   TRUE FALSE

   (c) You must always use `new` and `delete` when working with dynamic memory.

   TRUE FALSE

   (d) Polymorphism depends on dynamic typing.

   TRUE FALSE

   (e) `static_cast<>` provides a mechanism for compile-time type checking of casts.

   TRUE FALSE

   (f) `dynamic_cast<>` only works to cast between polymorphically-related types and void pointers.

   TRUE FALSE

   (g) Template definitions must be available to the compiler at instantiation time.

   TRUE FALSE

   (h) Name mangling is necessary for inheritance.

   TRUE FALSE

   (i) `extern "C"` tells the compiler to use C calling conventions.

   TRUE FALSE

   (j) C and C++ use different calling conventions, so it is not possible to link them together.

   TRUE FALSE

Extra Credit. (3 pts) Write a haiku about this class.

In case you're not familiar, a haiku is a poem in three lines, the first and third lines having five syllables, the second having seven. They're *supposed* to be profound. Here is an example:

Christmas is coming
The goose is already fat
Goose is expensive

and another:

This exam is done
I don't like the term "exam"
Please call it a test