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 << "Keep the change, ya filthy animal." << endl;
```

   (b)
```
string organization = "the American Dental Association";
cout << "Has this toothbrush been approved by " << organization << "?\n";
```

   (c)
```
string *s;

s = (string *) "This is my house.  I have to defend it.";
cout << *s << endl;
```

The next two problems use the function `name()`, defined below, with the output in (e) depending on the behavior in (d):

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

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

  return n;
}
```

(d) `name() = "Marv"`

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

The remaining problems depend on the class hierarchy defined below:

```
class staircase {
public:
  virtual void use(string name)
  {
    cout << name << " walks down the staircase" << endl;
  }
  virtual ~staircase() {}
};

class boobytrap : public staircase {
public:
  virtual void use(string name)
  {
    staircase::use(name);
    cout << name << " slips and falls" << endl;
  }
};
```

(f)
```
staircase s;
s.use("Kevin");
```

(g)
```
vector<staircase> v;
v.push_back(boobytrap());
v[0].use("Old Man Marley");
```

(h)
```
vector<staircase *> v;
v.push_back(new staircase());
v.push_back(new boobytrap());
v[1]->use("Harry");
```

2. (40 pts) Below is a simple, templated circular queue class. You will write code that works with this class (a method and an operator which is not a method) You may not alter the class definition, except as specified.

```
#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() {
  }
};
```

(a) (15 pts) Implement the copy constructor for `class circular_queue`. A copy constructor should always do a deep copy, when that is meaningful. The copy constructor is a member of the class, so you may assume that this code appears within the class definition on the previous page.

(b) (25 pts) Implement the overloaded output operator, `operator<<()`, for `class circular_queue`. **This operator is not a method of the class**, but—for convenience—all data in the class has been made public.

Your implementation should print the items in the queue from `front` to `back`, separated by commas. The entire set of these should be surrounded by a pair of curly brackets. You should not print any newlines within your operator implementation. For instance, given a queue of strings with values (`front` to `back`) "Kevin", "Harry", "Marv", "Peter", "Kate", your operator should print exactly the information within the quotes on the following line:

"`{Kevin,Harry,Marv,Peter,Kate}`"

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++:

   ```
   printf("Hello World!\n");
   ```

   TRUE FALSE

   (b) C++ is a superset of C.

   TRUE FALSE

   (c) Like C, C++ supports first class static dispatch.

   TRUE FALSE

   (d) Overloaded operators are actually functions.

   TRUE FALSE

   (e) You can use `print()` in C++, but you can also call `cout` to print.

   TRUE FALSE

(f) `free()` and `delete` are interchangeable.

TRUE FALSE

(g) References must be initialized at declaration time.

TRUE FALSE

(h) Initialization lists make it possible to call non-default constructors for instance variables.

TRUE FALSE

(i) Initialization lists make it possible to initialize reference instance variables.

TRUE FALSE

(j) This statement is true.

TRUE FALSE

Extra Credit. (3 pts) Write a haiku about this class. It doesn't have to be good. It doesn't even have to be a haiku (but that would be more fun). Just write something, and we'll give you three points.

Please avoid the words *segmentation*, its abbreviated form *seg*, *segfault*, *signal 11*, and *crash*. Kudos if you manage to make clear references to segmentation faults in 17 syllables without using any of these "illegal" words.

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:

When *Home Alone* saw
Its first theater showings
Teacher's mom took him

and another (with visual):

Did they really use
A torch on Joe Pesci's head?
Why'd he just stand there?

and a third:

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