Com S 327
Spring 2018
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	30	
2	40	
3	30	
EC	3	
Total	100	

1. (30 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.

   (a)
   ```
   cout << "Wiggle your big toe." << endl;
   ```

   (b)
   ```
   String name = "Hattori Hanzo";
   cout << "That really was a " << name << " sword." << endl;
   ```

   (c)
   ```
   string *s = (string *) "Ferraris...  Italian trash.";
   cout << *s << endl;
   ```

(d)
```
vector<string> v;
v[0] = "I should have been Black Mamba.";
v[1] = "O-Ren Ishii!  You and I have unfinished business!";
v[2] = "You must have big rats if you need Hattori Hanzo's steel.";

for (vector<string>::iterator i = v.begin(); i != v.end(); i++) {
  cout << *i << endl;
}
```

(e)
```
vector<string> v;
v.push_back(string("Which one do you wanna watch?"));
v.push_back(string("Shogun Assassin."));

for (vector<string>::iterator i = v.begin(); i != v.end(); i++) {
  cout << *i << endl;
}
```

(f)
```
try {
  throw "It's the wood that should fear your hand...";
}
catch (string s) {
  cout << s << endl;
}
```

2. (40 pts; 20 ea) Implement the methods specified given the following class. Assume that all methods are implemented—except for those which you are asked to implement—and work as expected (ask, if you're uncertain). You must implement the specified functionality fully within the assigned method; you may not alter the class declaration. An empty list is initialized with a null `head` and `tail`; otherwise, head addresses the first node in the list, and `tail` addresses the last.

```
template <class T>
class exam_list {
  class exam_list_node {
    public:
    T data;
    exam_list_node *next;
    exam_list_node *previous;
    inline exam_list_node(T d,
                          exam_list_node *n,
                          exam_list_node *p) :
      data(d), next(n), previous(p)
    {
      if (next) {
        next->previous = this;
      }
      if (previous) {
        previous->next = this;
      }
    }
  };
  private:
  exam_list_node *head;
  exam_list_node *tail;
  public:
  exam_list() : head(0), tail(0) {}
  ~exam_list();
  void insert_head(T d);
  void insert_tail(T d);

  // The code on the following pages is implemented here,
  // inside the class definition.
};
```

(a) Implement the overloaded assignment operator for `exam_list`. Also write the prototype in the specified location in the class definition. Your implementation should produce a deep copy.

(b) Implement the method `insert_sorted()` which inserts d in sorted order, smallest to largest, according to the comparitor `compare()`.

```
void insert_sorted(T d, int (*compare)(const T&, const T&))
{









}
```

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) The C compiler handles `extern "C"` declarations.

   TRUE FALSE

   (c) C++ does not allow name mangling.

   TRUE FALSE

   (d) Destructors for derived classes are called in the same order as the constructors.

   TRUE FALSE

   (e) Overloaded functions share both names and formal parameters.

   TRUE FALSE

   (f) Overloaded functions may not differ only in return type.

   TRUE FALSE

   (g) Function overloading requires name mangling

   TRUE FALSE

(h) Exceptions must be instances of `std::exception`.

TRUE FALSE

(i) You can compile any C program with a C++ compiler.

TRUE FALSE

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

TRUE FALSE

(k) Polymorphism is a static concept.

TRUE FALSE

(l) C and C++ use different calling conventions by default, but it's still possible to link them together.

TRUE FALSE

(m) C++ has first-class static dispatch.

TRUE FALSE

(n) C++ has first-class dynamic dispatch.

TRUE FALSE

(o) C++ has first-class double dispatch.

TRUE FALSE

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

For credit, your poem may not use any of the words *segmentation*, its abbreviated form *seg*, *segfault*, *signal 11*, or *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:

Summer time is near.
Amazon Lich Queens be damned.
Let's LARP in the park!

Okay, not so profound. Another:

A +3 poem!
I will wield it with vigor!
Sauron is a chump.

Also not profound. And a third:

Sunshine in the morn.
The bees fly from bloom to bloom.
Thunder storms at eve.

Woah! Mind blown!