Com S 229
Sprint 2015
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	1	
Total	100	

1. (30 pts; 5 ea) Give the output of the following code snippets, if any. Explicitly show newlines with a ↵. If the code does not produce output, write *no output*. If the code produces an error, write *error*. You may assume that all required headers are included and that the containing file uses the standard namespace. All parts of this problem are cumulative, meaning that functions and variables declared or assigned in one part are still alive, in scope, and retain their values in all later parts.

(a)
```
cout << "Strange women lying in ponds distributing swords "
     << "is no basis for a system of government." << endl;
```

Strange women lying in ponds distributing swords
is no basis for a system of government. ↵

(b)
```
cout << "It's a simple question of weight ratios!\n"
     << "A " << 5 << " ounce bird could not carry a "
     << 1 << " pound coconut.\n";
```

It's a simple matter of weight ratios! ↵
A 5 ounce bird could not carry a 1 pound coconut. ↵

(c)
```
stringstream ss;

ss << "Brave Sir Robin ran away." << endl;
ss << "Bravely ran away away." << endl;
```

no output

(d)
```
printf(&ss.str()[26]);
```

Bravely ran away away. ↵

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

```
const char *&tale() {
  static const char *knight = "Launcelot";

  cout << "The tale of Sir " << knight << ".\n";

  return knight;
}
```

Be vigilant, lest it bite your knee!

(e) `tale() = "Galahad";`

The tale of Sir Launcelot.↵

(f) `cout << (tale() = "Robin") << endl;`

The tale of Sir Galahad.↵
Robin↵

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 specified. 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.

```
class exam_list {
  class exam_list_node {
    public:
    const char *data;
    exam_list_node *next;
    exam_list_node *previous;
    inline exam_list_node(const char *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) {}
  // write prototype for 2a in the space below:
  exam_list(Const exam_list &el);

  ~exam_list();
  void insert_head(const char *d);
  void insert_tail(const char *d);
  // write prototype for 2b in the space below:
friend std::ostream &operator<<(std::ostream &o, Const exam_list &el)

};
```

(a) Implement the copy constructor for `exam_list`. Also write the prototype in the specified location in the class definition.

```
exam_list :: exam_list(const exam_list &el)
{
    exam_list_node *cur, *ccur;
    for (cur = el.head ccur = NULL; cur; cur = cur->next)
    {    ~~if (ccur != NULL)~~
         if (cur == el.head)
this.head = ccur = new exam_list_node(cur->data,
                                      NULL,
                                      ccur);

         else {
             ~~cur~~
             ccur -> next = new exam_list_node(cur->data,
                                               NULL,
                                               ccur)

             ccur = ccur -> next;
         }
    }
    this.tail = ccur;
}
```

(b) Implement a method to overload the output operator on class `std::ostream` for objects of type `exam_list`. Your implementation should write the information in all of the data nodes in the list, one node per line. Also write the prototype in the specified location in the class definition.

```
std::ostream &operator<<(std::ostream &o, const exam_list &el)
{
exam_list::exam_list_node *eln;
    for (eln = el.head ; eln ; eln = eln->next) {
        ~~std::cout~~
        ~~std~~ o << eln->data << std::endl;
    }
    return o;
}
```

3. (30 pts; 2 ea) Circle TRUE or FALSE in response to each of these statements about C++. You may additionally circle NOT SURE if you feel the need to indicate your lack of clarity, or write in a response for your amusement and ours. **To be clear, every statement is either true of false, so be sure to unambiguously circle exactly one of them per statement!** 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 = malloc(12 * sizeof (*i));
```

no cast

TRUE [✓ FALSE] NOT SURE WRITE-IN: ____________

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

TRUE [✓ FALSE] NOT SURE WRITE-IN: ____________

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

[✓ TRUE] FALSE NOT SURE WRITE-IN: ____________

(d) C++ supports first class dynamic dispatch.

[✓ TRUE] FALSE NOT SURE WRITE-IN: ____________

(e) C++ supports first class double dispatch.

TRUE [✓ FALSE] NOT SURE WRITE-IN: ____________

(f) `cout` is a function that you call to print to standard output.

TRUE [✓ FALSE] NOT SURE WRITE-IN: ____________

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

TRUE [✓ FALSE] NOT SURE WRITE-IN: ____________

(h) During its lifetime, a reference may refer to any number of variables.

TRUE [✓ FALSE] NOT SURE WRITE-IN: ____________

(i) Polymorphism depends on static typing.

TRUE [✓ FALSE] NOT SURE WRITE-IN: ____________

(j) `dynamic_cast<>` provides a mechanism for runtime type checking of casts.

[✓ TRUE] FALSE NOT SURE WRITE-IN: ____________

(k) Templates are instantiated with a type at runtime.

TRUE [✓ FALSE] NOT SURE WRITE-IN: ____________

(l) Name mangling is necessary for polymorphism.

TRUE [✓ FALSE] NOT SURE WRITE-IN: ____________

(m) Name mangling is necessary for function overloading.

[✓ TRUE] FALSE NOT SURE WRITE-IN: ______________________________

(n) To use an object instance in C code, simply call its methods.

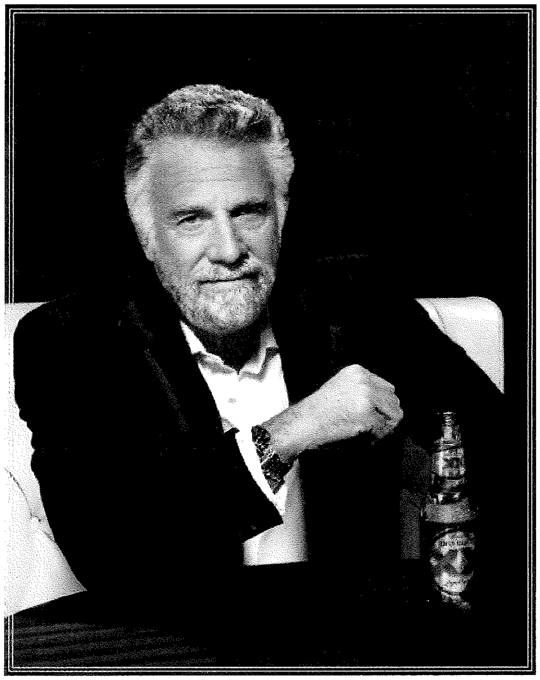
TRUE [✓ FALSE] NOT SURE WRITE-IN: ______________________________

(o) `const` is semantically equivalent in C and C++.

TRUE [✓ FALSE] NOT SURE WRITE-IN: ______________________________

Extra Credit. (1 pt) Complete the following.

I don't always program in C++



I write something!

But when I do. . .