CSE 333 Final Exam (2nd Midterm) 8/22/13

Name _____

There are 8 questions worth a total of 100 points. Please budget your time so you get to all of the questions. Keep your answers brief and to the point.

The exam is closed book, closed notes, closed electronics, closed telepathy, open mind.

If you don't remember the exact syntax for something, make the best attempt you can. We will make allowances when grading.

Don't be alarmed if there seems to be more space than is needed for your answers – we tried to include more than enough blank space.

Relax, you are here to learn.

Please wait to turn the page until everyone is told to begin.

Score _____ / 100

1. _____ / 20
2. _____ / 18
3. _____ / 18
4. _____ / 18
5. _____ / 12
6. _____ / 6
7. _____ / 6
8. _____ / 2 (!)

Question 1. (20 points) A little C++ hacking. On the following page, implement function `undup`. The input to this function is a list of strings that are not sorted in any particular order. The function should return a pointer to a newly allocated list of strings on the heap that is a copy of the original list, except that if the original list contains two or more adjacent copies of some string, those adjacent copies should be replaced by a single copy of that string. For example, if the input list contains

```
{"apple", "donut", "donut", "banana", "banana", "banana",
 "cherry", "cherry", "banana", "donut"}
```

then `undup` should return a pointer to a new `list<string>` containing the following:

```
{"apple", "donut", "banana", "cherry", "banana", "donut"}
```

Notice that a string might appear more than once in the result if it there are multiple copies of it in the original string separated by other strings.

For full credit, your function must scan the input list only once and may not use any other containers like lists, maps, or arrays. It may, of course, have whatever simple variables are needed, including strings.

Hints and reference information:

- Remember that a STL `list` is a linked list underneath, so you must use iterators to scan it; you can't use `[]` subscripts to access individual list elements.
- If `lst` is a STL `list`, then `lst.begin()` and `lst.end()` return iterator values of type `list<...>::iterator` that might be useful.
- If `it` is an iterator, then `*it` can be used to reference the item it currently points to, and `++it` will advance `it` to the next item, if any.
- Some useful operations on sequential STL containers, including `list`:
  - `c.clear()` – remove all elements from c
  - `c.size()` – return number of elements in c
  - `c.empty()` – true if number of elements in c is 0, otherwise false
  - `c.push_back(x)` – copy x to end of c
  - `c.push_front(x)` – copy x to front of c
- You are free to use the C++11 `auto` keyword, C++11-style `for`-loops for iterating through containers, and any other features of standard C++11, but you are not required to use these.

(Write your answer on the next page. You may remove this page from the exam if you wish.)

Question 1. (cont.) Complete the definition of function `undup` below. Some useful `#include` directives as well as a `using` directive have been provided for convenience. You should add any additional libraries or code that you need (although the sample solution only needed the ones given here).

```
#include <string>
#include <list>


using namespace std;

// return a pointer to a new heap-allocated copy of lst
// where adjacent duplicate strings in lst are replaced
// by a single copy of that string.
list<string> * undup(const list<string> &lst)  {









}
```

Question 2. (18 points) A little bit of class. Here is a tiny C++ class that holds a single integer as an instance variable.

```
class Int {
public:
  // constructors
  Int(): val_(0)    { cout << "cons "; }
  Int(int n)        { val_ = n; cout << "intcons "; }
  Int(const Int &v)
     : val_(v.val_) { cout << "copycons "; }

  // operations
  int get_val() {
    cout << "get_val ";
    return val_;
  }
  Int operator+(const Int &v) {
    cout << "op+ ";
    return Int(val_ + v.val_);
  }
  Int & operator=(const Int &v) {
    cout << "op= ";
    if (this != &v) { val_ = v.val_; }
    return *this;
  }
  Int & operator+=(const Int &v) {
    cout << "op+= ";
    val_ += v.val_;
    return *this;
  }

private:
  int val_;    // value stored in this Int
};
```

Answer the question about this class on the next page. You can remove this page for reference if you wish.

Question 2. (cont.) What output is produced when we execute the following program that uses the `Int` class defined on the previous page?

```
int main() {
  Int zero;
  Int one = zero + 1;
  cout << ";\n";
  Int two = zero;
  two += one + one;
  cout << ";\n";
  cout << two.get_val() + 1;
  cout << endl;
  return 0;
}
```

Output:

Question 3. (18 points) Virtual madness. What output is produced when we run the following program? It does compile and run without errors.

```
#include <iostream>
using namespace std;

class Base {
public:
  virtual void w() {      cout << "Base::w" << endl;   }
  virtual void x() { y(); cout << "Base::x" << endl;   }
          void y() {      cout << "Base::y" << endl;   }
  virtual ~Base()  {      cout << "Base::dtr" << endl; }
};

class A: public Base {
public:
  virtual void w() { x(); cout << "A::w" << endl;   }
          void y() {      cout << "A::y" << endl;   }
  virtual ~A()     {      cout << "A::dtr" << endl; }
};

class C: public A {
public:
  virtual void y() { cout << "C::y" << endl;   }
  virtual ~C()     { cout << "C::dtr" << endl; }
};

int main (int argc, char **argv) {
  Base *ptr = new C();
  ptr->y();
  cout << "-----" << endl;
  ptr->w();
  cout << "-----" << endl;
  ptr->x();
  cout << "-----" << endl;
  delete ptr;
  return 0;
}
```

Output:

Question 4. (18 points) Below is the pseudo-code for a very simple TCP server that accepts connections from clients and exchanges data with them. However, this code doesn't work because it has structural errors. In particular, some functions are called at the wrong time or in the wrong place, but there may be other problems. Write in corrections below to show how the pseudo-code should be rearranged, changed, or fixed to have the proper structure for a simple server. Feel free to draw arrows showing how to move code around, but be sure it is clear to the grader what you mean.

You should assume that all functions always succeed – ignore error handling for this question. Further, assume that the first address returned by `getaddrinfo` works and we don't need to search that linked list to find one that does work. Also, ignore the details of parameter lists – assume that all the "…" parameters are valid and appropriate.

```
int main(int argc, char **argv) {
  struct addrinfo hints, *rp;
  memset(&hints, 0, sizeof(hints));

  hints.ai_... = ...;   // specify values for options

  getaddrinfo(NULL, argv[1] &hints, &rp);

  // ok to assume *rp is a valid address and will work here
  int fd = socket(rp->ai_family,
                  rp->ai_socktype, rp->ai_protocol);

  setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, ...);

  freeaddrinfo(rp);

  while (1) {

    bind(fd, rp->ai_addr, rp->ai_addrlen);

    fd = accept(fd, ...);

    listen(fd, SOMAXCONN);

    // talk to client as needed
      read(fd, ...);
      write(fd, ...);

    close(fd);

  }
  return EXIT_SUCCESS;
}
```

Question 5. (12 points) Concurrency. Suppose we are executing a workload where each transaction requires 3 disk I/O operations with short bursts of CPU time at the beginning, at the end, and in between each pair of disk operations. A a single transaction (not drawn to scale) looks something like this:

CPU	I/O	CPU	I/O	CPU	I/O	CPU

Assume that each I/O operation takes 10 msec. (i.e., 0.010 sec.) and each burst of CPU time takes 10 microseconds.

(a) How many transactions per second can we perform if we execute transactions sequentially, where a new transaction can't start until the previous one is completed? You should give a "back-of-the-envelope" estimate – i.e., don't use a calculator (which is not allowed on the exam anyway). Just give a good answer accurate to a couple of significant digits. To help the grader, show enough of your work or give a brief justification so we can follow it.

(b) Now, suppose we use a multi-threaded implementation to execute as many transactions in parallel as possible. How many transactions per second can we execute now? You should assume that we have unlimited I/O resources so we can process as many I/O requests in parallel as desired, and you can assume that the concurrency does not add any measurable overhead or time needed to process the transactions.

Question 6. (6 points) One of the summer interns has been told to implement automatic memory management for our new implementation of Java. The idea was to implement a garbage collector to automatically reclaimed unused, dynamically-allocated data. But the intern thinks it would be simpler to use reference counting, where the implementation keeps track of how many pointers refer to each piece of dynamically allocated data, and frees (deletes) any data whose reference count becomes 0.

The question is, will this work? Is reference counting a suitable substitute for automatic garbage collection? Give a brief technical justification for your answer.

Question 7. (6 points) Almost all of the time when we are programming with classes and subclasses, we use virtual functions and dynamic dispatching so that the actual types of the data objects determine which functions are executed. But there are occasional times where it makes sense to omit the `virtual` keyword and determine the actual function at compile-time, regardless of the run-time types of the data. Give two (brief) reasons why we might want to do this.

(i)

(ii)

Question 8. (2 free points!)

What is the answer to this question? (Write an interesting answer below.)