University of Utah
School of Computing

CS 3500/5010 **Handout 11** **November 2, 2009**

Midterm Exam Solutions

The average grade on the midterm was 74 and the median grade was 75.

If you have any complaints about the way the exam was graded, write them on a separate sheet of paper, attach the paper to your exam, and give both to me by the end of the lecture on November 11. Do not modify your exam. We will then regrade your exam. If you do not follow this procedure, we will not regrade your exam.

(a) What is the key difference between black-box testing and white-box testing?

White-box testing takes implementation details into account; black-box testing deals only with the interface specification.

(b) What are two distinct advantages of using a version control system on a single-person project?

(1) It makes it easy to roll back a project to a previous state. (2) It makes it easy to keep the project consistent across multiple computers.

(c) In a program with two threads, how could a deadlock occur?

Suppose there are two locks, A and B. Thread 1 holds lock A and is waiting on lock B, and thread 2 holds lock B and is waiting on lock A. That's a deadlock.

(d) What are the five major phases in the software lifecycle?

1	analysis/requirements
2	design
3	implementation
4	testing
5	maintenance

(e) What is the major way in which statically and dynamically linked libraries are the same, and the major way in which they are different?

Both provide a way to integrate externally written object code into an executable. With static linking, the integration happens at link time. With dynamic linking, the integration happens at run time.

(f) At a minimum, how many of each of the following are needed at runtime to support the execution of a program with n threads?

Item	Quantity
Runtime stacks	n
Heaps	1
Program counters	n
Machine code versions of program	1
Processors	1

(g) For each of the programs identified below, indicate whether it is implemented using a client/server architecture.

Program	Yes or No
SVN	yes
Boggle	yes
Visual Studio	no
Random text generator	no
The Hunt	yes

(h) Suppose that you wish to construct a system that works in stages using the pipe and filter architecture. Its job will be to read input from `input.txt`, process it, and then write its results to `output.txt`. Your idea is to build the system from three programs (A, B, and C) by running them from the command line with:

```
A < input.txt | B | C > output.txt
```

From what source should each program be written to read its input and write its output? Answer this question by completing the following table. You'll need to write "cin" or the name of a file in each entry of column 2, and "cout" or the name of a file in each entry of column 3.

Executable	Input source	Output destination
A	cin	cout
B	cin	cout
C	cin	cout

(i) List five facilities provided by Visual Studio that aid in debugging programs.

1	breakpoints
2	single stepping
3	runtime stack display
4	inspection of variable values
5	watches and the list goes on

(j) Sockets provide a powerful abstraction. Like all abstractions, they allow a programmer to work at a higher level by selectively hiding details. What details do sockets hide from the programmer?

All of the low-level details of how computers communicate over a network.

The next four questions concern the class `Person`. This is `Person.h`:

```
#include <string>
#include <vector>
using namespace std;

// A Person object represents a person by recording his or
// her name and children.

class Person {

private:

    string name;                // Name of this person
    vector<Person> children;    // Children of this person

public:

    // Creates a Person with name n and no children
    Person (string n);

    // Adds a child to this Person's children
    void addChild (Person child);

    int mystery ();

};
```

This is part of `Person.cpp`:

```
#include "Person.h"
#include <algorithm>


// Simple helper function that invokes mystery on p.
int helper (Person &p) {
    return p.mystery();
}


int Person::mystery () {
    int n = children.size();
    vector<int> results(n);
    transform(children.begin(), children.end(), results.begin(), helper);
    int sum = 1;
    for (int i = 0; i < n; i++) {
        sum += results[i];
    }
    return sum;
}
```

(k) Provide the implementation for the `Person` constructor.

```
Person::Person (string n) {
    name = n;
}
```

(l) Provide the implementation for the `addChild` method.

```
void Person::addChild (Person child) {
    children.push_back(child);
}
```

(m) Provide a comment for the `mystery` method that describes what aspect of a `Person` the method calculates.

```
// Returns the number of descendants of this person (counting the
// person as one of his or her descendants).
```

(n) The `Person` class neither declares nor implements a destructor, copy constructor, or assignment operator. Will this cause problems, or will the default versions behave properly? Justify your answer.

This will not cause a problem. The member variables are both objects with their own destructors, copy constructors, and assignment operators that will be called automatically when a `Person` *object is destroyed, copied, or assigned. This default behavior will do the right thing.*

The next six questions concern the `Demo` class. Here is `Demo.h`:

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

class Demo {
private:
    int *numbers;
    int count;
public:
    Demo (int n);
    ~Demo ();
    int getNumber (int index);
    void setNumber (int index, int value);
    int getCount ();
};
```

Here is `Demo.cpp`:

```
#include "Demo.h"

Demo::Demo (int n) {
    count = 0;
    numbers = new int[n];
    for (int i = 0; i < n; i++) {
        numbers[i] = i;
    }
}

Demo::~Demo () {
    delete numbers;
}

int Demo::getNumber (int index) {
    count++;
    return numbers[index];
}

void Demo::setNumber (int index, int value) {
    numbers[index] = value;
}

int Demo::getCount () {
    return count;
}
```

Let `f`, `g` and `h` be defined as followed, and suppose that `h` is called.

```
void f (Demo *a) {
    a->setNumber(0, 40);
}


void g (Demo b) {
    a.setNumber(1, 50);
}


void h () {
    Demo* d = new Demo(5);
    f(d);
    cout << d->getNumber(0) << endl;   // Line A
    g(*d);
    cout << d->getCount() << endl;      // Line B
}
```

(o) Sketch the state of the stack and heap as they will be immediately before `f` returns. Draw carefully, label clearly, and write legibly.

(p) What is printed by Line A?

Answer	40

(q) Now sketch the state of the stack and heap as they will be immediately before `g` returns. Draw carefully, label clearly, and write legibly.

(r) What is printed by Line B?

Answer	1

(s) After `h` returns, how many bytes of memory will have been leaked? Assume that integers and pointers occupy 4 bytes, and that arrays and objects occupy space equal to the sum of the space occupied by their elements.

Answer	8

(t) In an attempt to solve the memory leak problem, suppose that `delete d;` is added as the last line of `h`. What specific problem will this cause?

The array that d contains was already deleted when `g` returned, and an attempt will be made to delete it again. This will cause a memory error.