Name: _____ UID: _____

CS 35L: Software Construction Lab

Section 3

Final Examination

Fall Quarter 2010

Monday December 6, 8:00 am

Time Limit: 2 hours and 45 minutes

Score: _____ out of 100

For each question in this exam packet, write your response legibly with a pencil in the allocated space. It may help to first formulate and write your answers on scratch paper, and then neatly copy them over. Printed and handwritten resources are permitted; electronic devices are not. Double check your answers if you have time afterwards. Raise your hand for question clarifications.

The following questions are worth 5 points each.

1. What's the difference between `sed` and `grep`? Be concise.

grep is used to match patterns, whereas sed is used to edit.

2. What does the following script print out?

```
1.  #!/bin/grep is
2.
3.  Hello
4.  this
5.  is
6.  a test!
```

this
is

3. What's a symbolic link (often referred to as a soft link)? How does this differ from a non-symbolic link (often referred to as a hard link)?

a sym link points to a file representation of the data (the name of the file) whereas a hard link refers to the actual physical data.

4. When you don't know how to do something in Linux, where should you first look (other than online)?

look at the man pages. you can also use whatis.

5. What's stored on a thread's (or process' for single threaded applications) stack?

local vars, function args, return addresses, old base pointers

The following questions are worth 7 points each.

6. As part of a class project you and a partner share access to an `svn` repository hosted on the class server. One night, without your knowledge, your partner adds a backdoor (a secret piece of code allowing remote control of the project) to the project repository. However, you do not realize this until after the project is turned in. How can you show (with reasonable certainty) that you did not add the back door to the project? List the specific commands you would use and what useful information they would provide.

just use git log, which will show you each commit as well as the author of that commit.

7. In 2009 an attack on SSL (Secure Socket Layer) Certificates was published online that allowed spoofing of domain names. This attack allowed a malicious site to publish valid security certificates which appeared to be for a different site. For example, www.goodsite.com.evilsite.com (a subdomain of evilisite.com) would claim to be www.goodsite.com by having a non-printable null character ("www.goodsite.com\0.evilsite.com") in its name. Given that every domain name is exactly 32 characters (with null bytes on the end when less than 32 characters are used), eliminate this spoofing vulnerability from the following function:

```
/*Returns 0 when the certificate is valid for the site and non-zero otherwise*/
int isValidForSite(char* domainName, char* certificateDomainName)
{
	return strcmp(domainName, certificateDomainName);
}
```

Write your corrected function below:

```
int isValidForSite (char *domainName, char* certificateDomainName) {
	for (int i = 0; i < 32; i++) {
		if (domainName == '\0') {
			for (int j = i; j < 32; j++) {
				if (domainName[j] != '\0') {
					return 1;
				}
			}
		}
		if (domainName[i] != certificateDomainName[i]) {
			return 1;
		}
	}
	return 0;
}
```

8. SSH is a widely used network protocol that allows for secure (in most typical situations) communication between two systems. SSH allows for both Public Key (where a client's public key is stored on the host) and Password based authentication which you saw in Lab 7. List a set of circumstances where Public Key authentication would be the most secure method and a set of circumstances where Password based authentication would be the most secure:

go public key based in the event that you're worried about a key logger on the system. use a password because public key is susceptible to someone impersonating a computer on the network that is authorized, even though they might not have a private key.

9. Which of the following bash commands would be most appropriate to compile a project and print a one line summary of the build status? Justify your answer by critiquing all three commands.

   a. make || echo "Build failed!" && echo "Build succeeded"

don't use this one because the ll operator will allow either make or the failure message to be printed, but the success message would be printed both ways.

   b. make && echo "Build succeeded" || echo "Build failed!"

use this because success only prints in the event that make returns with a success message, and will only print failed if make returns with a failure and success is not printed.

   c. make || echo "Build succeeded" && echo "Build failed!"

don't use this one because the ll operator will allow either make or the success message to be printed, but the failure message would be printed both ways.

10. Write a sed command that replaces every phone number in ./contact.html with its digit-only form. The format to consider is (ddd)-ddd-dddd. For example, (555)-555-5555 becomes 5555555555. Ignore phone numbers formatted incorrectly (for example, too many digits).

sed 's/[()-]//g' to remove () and - from the string.

The following questions are worth 10 points each.

11. You have a file called one_mb which is exactly 1 megabyte in size. You want to create from it a file of exactly 128 megabytes in size. Please write a shell script to do this with at most 9 lines and no loops, if statements, recursion, or any other logic control structures. Each command, including parameters, must be less than 100 characters in length.

1. cat one_mb one_mb > two
2. cat two two > four
3. cat four four > 8
4. cat 8 8 > 16
5. cat 16 16 > 32
6. cat 32 32 > 64
7. cat 64 64 > 128
8.
9.

12. In the following question, write your answers in Python.

a. Make a class Point3 that represents a 3-dimensional point. A constructor is optional, but you must implement the function distance2(…) that will compute the distance squared (e.g. $(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2$) between the current point and given point and return it:

```
class Point3:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z
    def distance2(self, x, y, z):
        return ((self.x - x)^2 + (self.y - y)^2 + (self.z - z)^2)
```

b. Write a function createCoordinates(…) that sequentially reads lines from a file in the form "x y z" (ex: "5 3 4", "2 -1 1") where all numbers are integers. The function should return an array of type Point3 containing all points read. The file name is a function parameter with default value "points.txt" (you can call createCoordinates(…) without a file name specified):

```
def createCoordinates (name=points.txt):
    a = [];
    with open(name) as f;
        for line in f:
            data = line.split()
            a.append(Point3(data[0], data[1], data[2]))
    return a
```

c. A bounding sphere is defined as a sphere which fully encloses a set of geometry. Create a function boundingSphere(…) that takes an array of type Point3 and prints out the coordinates of a bounding sphere's center followed by its radius that encloses all points in the array. You simply need *a bounding sphere* not the smallest possible bounding sphere. (Hint: This can be done with an arbitrary center point and a single pass of the array):

```
def boundingSphere(x):
    largestRad = 0
    for point in x:
        rad = distance2 (x.distance2(0,0,0))
        if rad > largestRad:
            largestRad = rad
    print "Center: (0,0), Radius: %d" % rad
```

13. The following C function counts the number of unique positive integer divisors a given number *n* has (not including 1 or *n*). For example, if the number 12 is given the function will return 4 since 12 can be divided by 2, 3, 4, and 6. Speed up this function by parallelizing it and doing all work in 3 child threads. Use `pthread_create` and `pthread_join` to manage your threads. You may add additional functions and global variables if needed. Do not attempt to improve the algorithm (i.e. still use an exhaustive search). You may assume that all operations are atomic, meaning you can perform operations such as `g_divisorCount++` without worrying about synchronization.

```
int g_divisorCount;

/* Returns 1 if the given number is prime, 0 otherwise */
int countDivisors(long long number)
{
	long long i;
	g_divisorCount = 0;

	for(i = 2; i <= number / 2; i++)
	{
		if(number % i == 0)
			g_divisorCount++;
	}

	return g_divisorCount;
}
```

Write your new function(s) below:

14. Given the following buggy C program to compute factorials (main.c):

```
 1.  #include <stdio.h>
 2.
 3.  int factorial(int n)
 4.  {
 5.    return n * factorial(n - 1);
 6.  }
 7.
 8.  int main()
 9.  {
10.    int n;
11.    scanf("%d", n);
12.    printf("Factorial of %n is", n, factorial(n));
13.
14.    return 0;
15.  }
```

And the following patch to correct the buggy program (called fix.patch):

```
 1.  --- main.c.orig       2009-12-04 04:50:04.000000000 -0800
 2.  +++ main.c     2009-12-04 04:51:44.000000000 -0800
 3.  @@ -2,14 +2,17 @@
 4.
 5.   int factorial(int n)
 6.   {
 7.  -  return n * factorial(n - 1);
 8.  +  if(n <= 1)
 9.  +    return 1;
10.  +  else
11.  +    return n * factorial(n - 1);
12.   }
13.
14.   int main()
15.   {
16.     int n;
17.  -  scanf("%d", n);
18.  -  printf("Factorial of %n is", n, factorial(n));
19.  +  scanf("%d", &n);
20.  +  printf("Fractorial of %d is %d\n", n, factorial(n));
21.
22.     return 0;
23.   }
```

Notice the typo of “Fractorial” on line 20 of the patch which should be “Factorial”. Please make a patch that can be applied to “fix.patch” and fix the typo (you do not need to use all lines) on the next page.

```
1.  —fix.patch.orig 2015-12-02 10:28:00.00000000 -0800
2.  +++ fix.patch 2015-12-02 10:28:00.00000000 -0800
3.  @@ -1,23 +1,26 @@
4.   --- main.c.orig 2009-12-04 04:50:04.000000000 -0800
5.   +++ main.c 2009-12-04 04:51:44.000000000 -0800
6.   @@ -2,14 +2,17 @@
7.    int factorial(int n)
8.   {
9.   - return n * factorial(n - 1);
10.  + if(n<=1)
11.  + return 1;
12.  + else
13.  + return n * factorial(n - 1);
14.  }
15.
16.  int main()
17. {
18.  int n;
19.  - scanf("%d", n);
20.  - printf("Factorial of %n is", n, factorial(n)); scanf("%d", &n);
21. - + printf("Fractorial of %d is %d\n", n, factorial(n));
22. + + printf("Factorial of %d is %d\n", n, factorial(n));
23.
24.  + return 0;
25. }
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
```

END OF EXAM