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 searches for a certain pattern from a file, or stdin if no file is specified.
sed, however, stands for stream editor and is mostly used to replace, delete or modify the text.

2. What does the following script print out?

```
#!/bin/grep is

Hello
this
is
a test!
```

The script outputs:
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 soft link is a file that has the information to point to another file/inode, which points to some data on the hard drive.
A hard link is a direct pointer to the original inode.
For example, renaming the original file that a hard link points wouldn't affect the hardlink. However, renaming the file that a soft link points to would decapacitate the soft link.

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

Either the man page or use the whatis command.
You can also use the '-h' or '--help' flag on commands. e.g.: wget -h

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

It's own sets of local variables, return addresses and frame 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.

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 isValideForSite (char * domainName, char * certificateDomainName) {
return (strlen(*domainName) == strlen(*certificateDomainName) &&
strcmp(domainName, certificateDomainName);
}
```

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:

In general, public key authentication is safer when you can assure that no one has the physical access to
your private key. Password based authentication would be safer in circumstances in which the private key can be accessed by physical means.

### Additional ###
Passwords are sent to server before verified, so the safety of it is relative to how well the server protects the transmission.

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"

If make succeeds, "Build succeeded" would never be echoed. If make fails, both "Build failed" and "Build succeeded" would be echoed. Not a good command to use.

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

Works perfectly.

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

If make successes, no message is printed. If make fails, both message would be printed.
Not a good command to use.

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" < input
```

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.
2.      cat one_mb one_mb > two_mb
3.      cat two_mb two_mb > 4_mb
4.      cat 4_mb 4_mb > 8_mb
5.      cat 8_mb 8_mb > 16_mb
6.      cat 16_mb 16_mb > 32_mb
7.      cat 32_mb 32_mb > 64_mb
8.      cat 64_mb 64_mb > 128_mb
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):
        self.x = 0
        self.y = 0
        self.z = 0

    def distance2(self, the_argument):
        dx = the_argument.x
        dy = the_argument.y
        dz = the_argument.z
        return (self.x - dx) ** 2 + (self.y - dy) ** 2 + (self.z - dz) ** 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 createCoordinate(path = "points.txt"):
    array = []
    with open(path, 'r') as f:
        while True:
            x = f.read(1)
            y = f.read(1)
            z = f.read(1)
            if not z:
                break
            point = Point3()
            point.x = x
            point.y = y
            point.z = z
            array.append(point)

    f.close()
    return array
```

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 (array):
   center = array[0]
   dist = 0
   for point in array[1:]:
      if center.distance2(point) > dist:
         dist = center.distance2(point)
   print("The coordinate of the center is" + center.x + center.y + center.z
+ "the radius is" + dist)
```

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;
}
```

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.

Write your new function(s) below:

```
#include <pthread.h>
#include <stdio.h>

#define NUM_ELEMENTS 4000000000
#define NUM_THREADS 4
int a[NUM_ELEMENTS];

pthread_t th[NUM_THREADS-1];

struct info{
  int id;
  int interval;
  int res;
};

void* t(void* arg)
{
  int thread_num=(int)arg;
  int i=0;
  int step=NUM_ELEMENTS/NUM_THREADS;
  int start=step*thread_num;
  int end=start+step;
}
```

```
int main(void)
{
  int i;
  for(i=0;i<NUM_THREADS-1;i++)
    pthread_create(&th[i],NULL,t,(void *) i);

  t((void*)0);
  for(i=0;i<NUM_THREADS-1;i++)
    pthread_join(th[i],NULL);

}
```

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

```
#include <stdio.h>

int factorial(int n)
{
  return n * factorial(n - 1);
}

int main()
{
  int n;
  scanf("%d", n);
  printf("Factorial of %n is", n, factorial(n));

  return 0;
}
```

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 <u>next page</u>.