SENG 265: Software Development Methods
Fall 2012

Midterm Exam
October 25, 2012

Student name:	_____
Student number:	_____
Marks:	_____

Note:

- There are seven (7) pages in this exam paper containing twelve (12) questions. Please count the pages in your copy of the exam *and immediately notify the instructor if a page is missing*.
- All answers are to be written on this paper.
- This is a 50-minute exam.

Section A (30 marks): 10 questions

For each question in this section, **you must clearly circle all answers** that apply. No partial marks are awarded for a question. All questions have equal weight.

Question 1: One of your current projects is under Subversion control, and you have a working copy in your account. Members of the project team have fixed some bugs and committed their changes to Subversion. In order to ensure your working copy is now consistent with the changes in the repository, you must:

a. Perform an *svn checkout* on the newly changed files.
b. Perform an *svn add* on the newly changed files.
c. Use *sftp* or scp to transfer files from the repository into your working copy.
d. Delete your working copy and then have a team member perform *svn commit* right away.
e. None of the above.

Question 2: With respect to *svn add*:

a. You may perform this command on files and directories.
b. You must be careful as it supposed to change the permissions of the files in your working copy of the project.
c. It must be done before a new file can be committed to the repository.
d. It will automatically update files inside the directory where the command is executed.
e. None of the above.

Question 3: The term *copy-modify-merge* refers to:

a. A style of using Subversion that is strongly discouraged.
b. An approach to handling conflicts.
c. An approach towards debugging programs with multiple files.
d. A tool that merges changes into C files automatically via an option to *gcc*.
e. None of the above.

Question 4: The bash shell's *history* command:

a. Sets the *HISTORY* environment variable to the value of the command's argument.
b. Produces output that may vary from user to user.
c. Can be provided a number as an argument.
d. Produces output that may vary from login session to login session.
e. None of the above.

Question 5: To set the permissions for the directory "/home/finnandjake/.www/index.html" to be *readable and writeable* by the user, *readable* by group members, and *inaccessible* to all others, we would use the command:

a. chmod 755 /home/finnandjake/.www/index.html
b. chmod u=rw,g=r,o=r /home/findandjake/.www/index.html
c. chmod 770 /home/finnandjake/.www; chmod go-wx /home/finnandjake/.www
d. chmod u+rwx /home/finndandjake; chmod go=rx /home/findandjake/.www/*
e. None of the above.

Question 6: A UNIX piped command (*i.e.*, two commands connected with the "|" symbol) is:

a. A way of connecting one command's *stdin* to another command's *stdin*.
b. Used to put all of the commands to sleep.
c. Different than what is achieved by output redirection (*i.e.*, the ">" symbol).
d. A way of connecting one command's *stdout* to another command's *stdin*.
e. None of the above.

Question 7: Assuming we have declarations "`char command[50]; char *cp;`" in our C program, then which expressions have the same value type on either side of the assignment operator?

a. cp = &command[5];
b. *cp = command[5];
c. *cp = &command[5];
d. cp = command[5];
e. None of the above.

Question 8: Consider the following nested loops in C found within some function:

```
int i, j;
for (i = 0; i < 5; i++) {
    for (j = 0; j < 5; j++) {
        if (j < i) {
            printf("*");
        } else {
            printf("X");
        }
    }
    printf(" ");
}
printf("\n");
```

The output seen on the terminal when this code is executed (where a printed space is represented by "␣") is:

a. *****␣X****␣XX***␣XXX**␣XXXX*␣
b. XXXXX␣*****␣XXXXX␣*****␣
c. XXXXX␣*XXXX␣**XXX␣***XX␣****X␣
d. (program does not compile)
e. None of the above.

Question 9: The "continue" statement in C can be used:

a. to transfer control directly out of a loop and return from the function enclosing the loop.
b. to transfer control from a loop to the last statement in the loop.
c. to transfer control from an *if* statement to the first statement after the *if* block.
d. to transfer control from a statement in a *while* block to the start of the loop by executing the loop's conditional expression.
e. None of the above.

Question 10: According to what has been taught in this course, tuples and lists in Python:

a. can have their elements accessed in a style similar to array indexing in Java.
b. are simple data structures available to programmers.
c. are immutable when used in a *for* loop.
d. are mutable when used in functions.
e. None of the above.

Section B (20 marks): One question

Question 11: Consider an array of random integers:

```
int[] arr = { 31, 41, 59, 2, 653, 5, 89, 323, 8, 4, 6,
              26, 43, 38, 3, 27, 950, 2, 88, 419, 79, 68, 399, 3, 0 };
```

The final "0" indicates there are no more numbers to follow in the array. (Note that the number zero will never be a part of an array of integers other than to indicate the end of the array.)

Write a C function accepting such an array as a parameter and returning:

- the minimum value
- the maximum value
- the average value
- the number of elements in the array.

Moreover the values must be returned in a C struct, and the struct is to be of your own design. You must provide all code for the function and for the structure declaration. Some marks will be given for the quality of your solution.

(room for answer to question 11)

Section C (30 marks): One question

Question 12

Write a C function named "in_string" that accepts two strings as parameters and returns an integer whose value is either:

- -1 if the second string is not a substring of the first, or
- the index position in the first string at which the second string appears for the first time as a substring.

For example, below are five sample calls with their corresponding return values:

- in_string("moose", "deer") *returns -1*
- in_string("furball", "fur") *returns 0*
- in_string("furball", "ball") *returns 3*
- in_string("furball", "balls") *returns -1*
- in_string("furfurballball", "ball") *returns 6*

You may not use *strstr()* or any of the string-library routines. You may assume the second string is always shorter than the first, and that neither string will ever be the empty string. You must not hardcode the strings (*i.e.*, your solution must work for any pair of strings, and not just those given above as examples).

Some marks will be given for the quality of your solution.

(room for answer to question 12)