University of Victoria

Examinations December 2005

SENG265 F01,F02 – Software Development Methods

Name: _____ Student No. _____

Instructor: Michael Zastre Section: _____

Duration: 3 Hours

To be answered on the paper.

Students must count the number of pages in this examination paper before beginning to write, and report any discrepancy immediately to the invigilator.

This question paper has 11 pages.

- You must obtain permission from an invigilator to leave the examination room temporarily.
- No cell phones are permitted.
- This is a closed-book, closed-notes exam.
- Write legibly.

Part	Mark
A	
B	
C	
Total (100 max)	

Part A: Multiple Choice [Questions 1 to 20 worth one mark each]

For each question, *circle all answers that apply*. Some answers may have more than one correct answer – **you must circle all correct ones**. The language / system referred to by the question appears in square brackets.

1. [C] The "malloc()" function:
   (a) is used to obtain memory from the stack for variables.
   (b) is used to obtain memory from the stack or heap for variables.
   (c) returns a class instance.
   (d) returns a memory address.

2. [C] Functions in the C language:
   (a) must be defined before they are used.
   (b) may access global variables.
   (c) use call-by-value parameter passing.
   (d) may be nested within each other.

3. [C] The variable declaration "`int *a[10];`" means that:
   (a) `a` is a pointer to an array of ten integers.
   (b) every element in the array `a` is a pointer to an integer.
   (c) `a[10]` is a pointer to an integer.
   (d) `a` stores ten memory addresses.

4. [C] After the function call `strncpy(s1, s2, len)` is executed:
   (a) a duplicate of `s1` will be made and stored in `s2`.
   (b) `len` bytes will be obtained from the heap.
   (c) up to `len` bytes will be copied.
   (d) `s1` may be the empty string.

5. [C] When a segmentation fault occurs at runtime, then:
   (a) we press "enter" to continue program execution.
   (b) we can use a debugger to find the statement generating the fault.
   (c) the executable file was erased and we must recompile the code.
   (d) the source was erased and we must obtain a new copy from CVS.

6. [C] "Function prototypes":
   (a) are used to indicate the types of parameters and return values of functions.
   (b) are partial implementations of function bodies.
   (c) must agree with types used for corresponding function definitions.
   (d) are commands given to the debugger.

7. [C] Strings may be concatenated together with the:
   (a) use of the "+" operator.
   (b) use of the "." operator.
   (c) use of the UNIX "cat" command.
   (d) help of string library functions.

8. [C] Structures in C:
   (a) are another form of arrays.
   (b) are a heterogeneous aggregate type.
   (c) may contain other structures.
   (d) can be allocated statically or dynamically.

9. [C] When accessing elements of an array in C, we:
   (a) can depend upon a "segmentation fault" if we access an invalid index..
   (b) must index from 1.
   (c) can easily obtain the memory address of each element.
   (d) iterate through the array using the "foreach" operation.

10. [C] To construct a string representation of some integer:
   (a) use the "`itoa()`" function.
   (b) use `sprintf`.
   (c) we could write our own function to do this.
   (d) cast the integer to a string via the use of `(char *)`.

11. [Perl] Perl programs are "scripts", which means:
   (a) they are compiled by Perl into an executable and then run.
   (b) they need a "filesystem path" at the start to indicate script location.
   (c) they are text files directly interpreted as programs.
   (d) special keywords must be indicated by using a different font.

12. [Perl] The "`system()`" function:
   (a) is used to execute shell commands from a Perl script.
   (b) accepts a hash array of commands as an argument.
   (c) returns the number of commands executed.
   (d) if used, must appear as the first statement in a script.

13. [Perl] The "`my`" keyword:
   (a) indicates the start of a subroutine.
   (b) can be used in global and local scopes.
   (c) produces the same effect as the "`local()`" keyword.
   (d) can be used to distinguish two variables having the same name.

14. [Perl] Associative arrays (i.e., variables with names beginning "%"):
    (a) map string "keys" to pure integer "values".
    (b) always store keys in alphabetical order.
    (c) may be assigned an initial mapping of keys to values.
    (d) can also called "hashes", "hash arrays" or "hash tables".

15. [Perl] Perl subroutines:
    (a) Can accept an arbitrary number of scalar-variable parameters.
    (b) Cannot return lists.
    (c) Cannot use global variables.
    (d) Can call any other subroutine or function, including itself.

16. [UNIX] From the current directory, obtain a listing of all text files in the "`src`" subdirectory, storing the output in the original current directory.
    a. `cd src; ls *.txt > listing.txt; cd ..`
    b. `ls *.txt > listing.txt`
    c. `ls src/* | grep "*.txt" > listing.txt`
    d. `cd src; ls *.txt > ../listing.txt`

17. [UNIX] Change the permissions for user "`asimpson`" such that their `.www` directory is world readable and executable.
    a. `chmod ~/asimpson/.www rw`
    b. `cd ~asimpson; \rm .www; mkdir .i_luv_big_macs; chmod go+rw .www`
    c. `chmod /home/asimpson/.www o+rw`
    d. `chmod world+rx ~asimpson/.www`

18. [UNIX] Start the "`deathstar`" program running in the background, with program output sent to Darth Vader's (user: `dvader`) file "`dstar.output`".
    a. `deathstar &; > ~dvader/dstar.output`
    b. `& deathstar > ~/dvader/dstar.output`
    c. `deathstar > ~dvader/dstar.output &`
    d. `(deathstar | cat > ~dvader/dstar.output) &`

19. [Software Engineering Context] According to our lectures, the first 3 phases of the software lifecycle can be named:
    (a) requirements, recording, design
    (b) exchange, feedback, design
    (c) exchange, breakdown, design
    (d) requirements, specification, design

20. [Software Engineering Context] When referring to "intellectual control" of a software development process, we mean that:
    (a) complexity makes the design process easy.
    (b) the mind can grasp any amount of detail and still grasp the situation.
    (c) eliminating irrelevant detail aids abstraction.
    (d) natural language introduces ambiguity, which we seek to eliminate.

Part B. Perl programming [20 marks total]

21. [10 marks] Consider the following Perl script and the input provided below in the box. What is the output of the script? Use as many output lines as needed. The following code contains no syntactic errors, produces no warnings, and does print results.

```
#!/usr/bin/perl
use strict;

while (<>) {
        chomp $_;
        my @in = split (/ /, $_);
        my $a = 1;
        my $b = 2;
        if (@in == 2) {
                my $a = shift @in;
                my $b = shift @in;
                print "$a:$b:", (($a + $b)/2), "\n";
        } elsif (@in >= 4) {
                my $a = shift @in;
                my $bias = shift @in;
                my $val = shift @in;
                for my $m (@in) {
                        $val = $val + ($a * $bias);
                        print $m, ":";
                }
                print $val, "\n";
        } else {
                my $m = shift @in;
                print "$m:$b\n";
        }
}
```

```
1 2
3 4 5
6 7 8 9
1 2 3 4
```

18. [10 marks] Write a Perl script that accepts two lines of integers from the console and prints their dot-product on the console. For example:

```
3 2 1
10 1 2
```

will produce the output:

```
34
```

You may use regular expressions; you may not use references. Assume that the number of integers on each line is the same, and each number is separated by a single space, but make no assumptions about the number of such integers.

Part C: C programming language [60 marks total]

19. [5 marks] What is the output of the following program? Write your answer in the lines provided below.

```
1     #include <stdio.h>

2     int c = 3141;

3     int mystery(int a, int b)
4     {
5             int temp;
6             temp = a;
7             a = b;
8             b = temp;

9             return temp;
10    }

11    int main()
12    {
13            int x = 59;
14            int y = 26;
15            int z = 53;
16            printf("%d %d %d %d\n", c, x, y, z);
17            c = mystery(x, y);
18            printf("%d %d %d %d\n", c, x, y, z);
19            c = mystery(z, c);
20            printf("%d %d %d %d\n", c, x, y, z);
21            return (0) ;
22    }
```

Output

_____

_____

_____

_____

20. [10 marks] What is the output of the following program? Write your answer in the lines provided below.

```
1     #include <stdio.h>

2     int main()
3     {
4             int a[3] = {11, 22, 33};
5             int b = 1000;
6             int *p1 = &b;
7             int *p2 = a;
8             int *p3 = &a[2];

9             printf("%d %d %d %d %d %d %d\n", a[0], a[1], a[2],
                  b, *p1, *p2, *p3);

10        *p2 = 44;
11        p3 = p1;
12        *p3 = 55;
13        printf("%d %d %d %d %d %d %d\n", a[0], a[1], a[2],
                  b, *p1, *p2, *p3);

14        a[2] = 99;
15        *p3 = 101;
16        printf("%d %d %d %d %d %d %d\n", a[0], a[1], a[2],
                  b, *p1, *p2, *p3);

17        *(p2 + 1) = 555;
18        a[0] = 777;
19        printf("%d %d %d %d %d %d %d\n", a[0], a[1], a[2],
                  b, *p1, *p2, *p3);
20        return (0);
      }
```

Output

21. [25 marks] Complete the C function described below. Some marks will be given for the quality of your code (i.e., code which anticipates possible errors, etc.). List all assumptions; note that unreasonable assumptions will be penalized.

```
/*
 * capitalize_word
 *
 * input:
 * -- char *s: a string of words
 * -- char *t: a string representing a single word
 *
 * purpose: all occurrences of string t in string s are
 *    converted to upper-case letters. The original string
 *    s is not modified; the function returns a copy of the
 *    modified string. Case is to be ignored. Assume that
 *    words are separated by a single space. Hint: use
 *    "toupper()" and "tolower()" as needed.
 *
 * example: if s = "This is not that, nor is this the thistle." and
 *    t = "this", then the string returned is:
 *    "THIS is not that, nor is THIS the thistle.".
 *

char *capitalize_wordy (char *s, char *t)
```

(blank page)

END