University
of Victoria

University of Victoria
Final Examination
December 2011

Course Name & No.	SENG 265 Software Development Methods
Section(s)	A01 and A02
CRN	10785 and 15390
Instructor	Michael Zastre
Duration	Three (3) hours

Name:
Student Number: V00 _____

This exam has a total of 16 pages including this cover page.

Students must count the number of pages and report any discrepancy immediately to the Invigilator.

This exam is to be answered on the paper.

No materials or aids of any kind are permitted. This is a closed-book exam.

Ensure all cellphones are switched off. You must obtain permission from an invigilator to temporarily leave the examination room.

The total number of marks in this exam is 180.

Section A (40 marks): 20 questions

For each question in this section, **clearly circle all answers** that apply. All questions have equal weight. Partial marks are not given for incomplete answers.

Question 1: We can say that variables in Python programs:

a. Are always typed.
b. Are always global.
c. Hold references to memory that is garbage collected.
d. Must always store the same type of value after are they are first assigned.
e. None of the above.

Question 2: Methods (or member functions) of a Python class:

a. May have the "`private`" keyword associated with the definition to ensure the function is only ever accessed from within the class.
b. Can be called without passing the object's reference as an argument to the function.
c. Can be defined without referring to the object's reference in the function signature.
d. Can have local variables that are not instance variables.
e. None of the above.

Question 3: A Python class:

a. May inherit from other Python classes in an object-oriented style.
b. Can have class variables.
c. Can have private variables.
d. Can be stored in a Python script that defines other classes.
e. None of the above.

Question 4: A Python list variable:

a. May be indexed as an array.
b. Can have elements added to the front and to the end.
c. Must have elements that are all the same type.
f. Can be made immutable with the "`private`" keyword.
d. None of the above.

Question 5: The Python statement "`for i in range(lower, upper): print i`"

a. Sets the value of "`upper`" to "`lower`".
b. Sets the value of "`lower`" to "`upper`".
c. Calls the function "`range`" which returns the string "`abcde`" when lower is "a" and upper is "e".
d. Will never compile because the "`for`" operator should be replaced with a "`while`" when "`range`" appears in an expression.
e. None of the above.

Question 6: Any C source-code file:

a. Must contain the "`main()`" function as otherwise the file cannot be part of a C program.
b. Can be compiled on its own to produce an object file (i.e., "`.o`" file).
c. Must include function prototypes for all other functions in all other modules.
d. Can have global variables that are only visible within the file and not outside of it.
e. None of the above.

Question 7: The C string function "`strncpy(foo, bar, baz)`":

a. Takes the string in "`bar`" and copies it to "`foo`".
b. Ensures the null character is always copied into the new string.
c. Automatically allocates memory for the new string.
d. Is safer to use than "`strcpy`".
e. None of the above.

Question 8: Dynamic memory associated with a variable "`char *bar;`" in function "`foo()`":

a. May be allocated using `malloc()` or a related function.
b. May be deallocated using the `delete()` operation.
c. Will be inaccessible after `foo()` returns if "`bar`" is itself a local variable.
d. Can be accessed by buggy code in other parts of the program that don't refer to "`bar`".
e. None of the above.

Question 9: ANSI C function prototypes:

a. Are pseudocode versions of the functions they represent.
b. Are another name for the "`#include`" lines.
c. Contain function names and parameter types.
d. Need not match their corresponding function definitions in the same file.
e. None of the above.

Question 10: "Separable compilation" means:

a. All modules must be compiled simultaneously to build an executable.
b. A "`makefile`" is required before compilation can even begin.
c. Changes to one module do not necessarily result in recompiling all others.
d. Compilation separates a ".c" file into a ".o" file and a ".h" file.
e. None of the above.

Question 11: Pipes and input/output redirection are important tools available via the UNIX shell. Which of these UNIX commands properly ensures the output of `step_a.sh` is piped into `process_b.py`, and that `process_b.py`'s output is saved in `/tmp/capture.txt`? Assume both of the scripts/programs are executable. (Remember to circle all that apply.)

```
a. cat ./options.txt | step_a.sh | process_b.py > /tmp/capture.txt
b. cat ./options.txt > step_a.sh > process_b.py > /tmp/capture.txt
c. cat ./options.txt | step_a.sh | process_b.py | /tmp/capture.txt <
d. cat ./options.txt | step_a.sh | process_b.py  || /tmp/capture.txt
```

e. None of the above.

Question 12: The UNIX "`ls`" command:

a. Is meant to accept input that can be redirected to its `stdin` from a command like "`cat`".
b. Can display the contents of files.
c. Can display the attributes of files.
d. Can display the size of files.
e. None of the above.

Question 13: The UNIX "`history`" command:

a. Displays a list of all previous logins to your account.
b. Displays a history of all files created or destroyed in your account.
c. Displays a history of errors and segmentation faults that occurred in your shell.
d. Displays a list of recent commands used in your shell.
e. None of the above.

Question 14: After executing "`svn add blarg.c`", the user who issued this command:

a. Knows the file named "`blarg.c`" is now known by the repository.
b. Knows the file named "`blarg.c`" has been committed to the repository.
c. Has created a new file in their working module.
d. Has taken a step that may allow another user to access the contents of "`blarg.c`".
e. None of the above.

Question 15: The command "`svn checkout http://somesite.org/somemodule`":

a. Can only be performed once for "`somemodule`".
b. Will return an error if some other user has already checked out "`somemodule`".
c. Creates possibly many subdirectories.
d. Is the opposite of "`svn checkin http://foo/somemodule`".
e. None of the above.

Question 16: A bug is:

a. Evidence the programmer might misunderstand the meaning of their code.
b. Can always be found using "`printf`" statements in C code.
c. Best found using a methodical, scientific approach.
d. An error caused by the compiler.
e. None of the above.

Question 17: "Black box" testing is the same as:

a. "Functional" testing.
b. Testing of Python scripts.
c. "Structural" testing.
d. "Blue box" testing (i.e., for checking the correctness of garbage-collector implementations).
e. None of the above.

Question 18: "Execution-based" testing assumes that:

a. The language always has a unit-test framework available for it.
b. Test cases and data are generated by executable code.
c. A completed detailed specification is available that describes the computer program.
d. Parts of the program are available that can be compiled and run.
e. None of the above.

Question 19: Manual testing techniques (such as Fagan inspections):

a. Assume the participation of very experienced programmers as inspectors of the code being tested.
b. Assume the participation of the programmer who wrote the code, who then defends that code as does a defendant at a court trial.
c. Assumes the existence of a checklist of well-known faults.
d. Assumes the existence of a moderator who organizes the inspection and chairs the inspection session.
e. None of the above.

Question 20: A "makefile":

a. Is always a shell script.
b. Will only work as a program if the first line is "`import unittest`".
c. Requires a special "make" editor in order to be modified.
d. Can be used to build an executable.
e. None of the above.

Section B: Python (50 marks)

Question 21 (weight 10): Consider the following Python code, named "mystery.py":

```
#!/usr/bin/python
import sys

def main():
    (aaa, bbb, ccc) = (0, 0, 0)

    ddd = sys.stdin.readlines()

    while (ddd != []):
        a_ddd = ddd[0]
        ccc = ccc + 1
        bbb = bbb + len(a_ddd)
        a_ddd = a_ddd.strip()
        words = a_ddd.split()
        aaa = len(words) + aaa
        ddd = ddd[1:]
        print aaa, bbb, ccc

if __name__ == "__main__":
    main()
```

and the following input, named "in01.txt":

```
this is the time
for all good
people to come
to the aid of their country.
```

Write on the lines below the output that results when the input is redirected into the Python program (i.e., "`cat in01.txt | ./mystery.py`").

Question 22: Weight 30

Write a Python function called `compute_hlav()` for analyzing a file containing stock-price or index-price data. The function will accept three parameters: the name of the file, a starting date in the form YYYYMMDD, and an ending date in the form YYYYMMDD. The function will return the highest price, lowest price, and average daily volume indicated by the data. Each line in the file is formatted as "`date open high low close volume adjusted_close`". For example, if the file "`GE.txt`" contains the following five lines:

```
20090312    8.73    9.78    8.43    9.57    355002000    9.44
20090313    9.9     9.98    9.3     9.62    216287500    9.49
20090311    9.23    9.36    8.31    8.49    251329800    8.37
20090316    9.97    10.36   9.6     9.66    226193300    9.52
20090310    8.01    8.99    7.95    8.87    368705700    8.75
```

and if the following line appears in a script using your function:

```
(high, low, average_volume) = compute_hlav("GE.txt", "20090311", "20090313")
```

then `high` equals 9.98, `low` equals 8.31, and `average_volume` equals 274206433. Assume all date values are legal.

Hints: Use "`int(somestring)`", "`float(somestring)`" to convert a string to an integer and float (respectively). Use "`str(someval)`" to covert a integer or floating point value to a string. Use "`input = open(filename, 'r')`" to open a file named "`filename`" and store the file pointer in "`input`". Make proper use of tuples and lists.

Your indentation must be clearly indicated. Some marks will be given for the quality of your answer.

Question 23: Weight 10

Python strings and tuples are referred to as *immutable*, and lists are referred to as *mutable*. What is the difference between immutable and mutable variables, and what are the consequences for the Python code that you write? Provide examples.

Section C: C programming (40 marks)

Question 24: Weight 15

What is the output of the following program? Write your answer in the lines provided below.

```
#include <stdio.h>

void process (int *m, int *n, int r, int s)
{
    *m = r;
    *n = s;
}

int main()
{
    int p[4] = {15, 25, 30, 55};
    int b = 2000;
    int *p1 = &b;
    int *p2 = p;
    int *p3 = &p[3];

    printf("%d %d %d %d %d %d\ %d %dn", p[0], p[1], p[2],
        p[3], b, *p1, *p2, *p2, *p3);

    process (p1, p3, 3000, 4000);
    printf("%d %d %d %d %d %d %d\n", p[0], p[1], p[2],
        p[3], b, *p1, *p2, *p2, *p3);

    p[0] = 4321;
    process (&p[3], &p[2], p[1], p[0]);
    printf("%d %d %d %d %d %d %d\n", p[0], p[1], p[2],
        p[3], b, *p1, *p2, *p2, *p3);

    *(p2 + 3) = 1234;
    p[0] = 3333;
    printf("%d %d %d %d %d %d %d\n", p[0], p[1], p[2],
        p[3], b, *p1, *p2, *p2, *p3);

    exit(0);
}
```

(Your output must appear below.)

Question 25: Weight 25

Complete the C function describe below. Some marks will be given for the quality of your code (i.e., dealing with possible error conditions). List all assumptions; note that unreasonable assumptions will be penalized. (Note that the input parameter must not be modified by your solution.)

```
/*
 * shrink_array
 *
 * input:
 * -- int *a: an existing array of integers, where the last element is
 *            indicated by a negative value. The contents of this array must
 *            not be overwritten.
 *
 * purpose: each positive, non-zero value in a[] is copied into b[].
 *    the new array is returned.
 *
 * example: if a[] = {3, 0, 4, 0, 5, 31, -1} then the value returned
 *     by shrink_array is some integer array of the value {3, 4, 5, 31, -1}
 */

int *shrink_array(int *a)
```

Section D: Testing (20 marks)

Question 26: Weight 20

You are asked to test the code for a "phonelist" application on a cellphone. Each entry in the phonelist has a name, phone number, and ringtone. Given the memory constraints of the cellphone, the developers must code this application using a special flavour of embedded C. The designers have told you that up to 256 phonelist entries can be stored in the cellphone, and a further 128 entries can be stored in the phone's SIM card if there are more entries than can be stored in the cellphone itself.

Using equivalence classes and boundary-value analysis, derive and describe a list of test cases for the phonelist application.

Section E: Make (10 marks)

Question 27: Weight 10

Consider the following makefile:

```
CC=gcc
FLAGS=-ansi -pendantic -Wall

parser: parse.o line.o pstack.o
	$(CC) -o parser line.o pstack.o parse.o

parse.o: parse.c token.h line.h pstack.h
	$(CC) $(FLAGS) -c parse.c

line.o: line.c line.h token.h
	$(CC) $(FLAGS) -c line.c

pstack.o: pstack.c pstack.h token.h
	$(CC) $(FLAGS) -c pstack.c

clean:
	rm -f *.o parser
```

(a) What must be recompiled if `token.h` is the only file changed since the last build?

(b) What must be recompiled if `parse.h` is the only file changed since the last build?

(c) Why do you think nothing follows immediately after "`clean:`"?

(d) How many rules does this makefile contain?

(e) Explain why a makefile should be used for a non-trivial project.

Section F: Miscellaneous (20 marks)

Question 28: Weight 10

What are some of the techniques we can use to debug our program when we have "good clues"?

Question 29: Weight 10

Explain the difference between "black-box testing" and "white-box testing". Use an examples.

END

This page is for the sole use of examination evaluators.

Section A	/40
Section B	/50
Section C	/40
Section D	/20
Section E	/10
Section F	/20
Total	**/180**