University of Victoria

University of Victoria
Final Examination
April 2013

Course Name & No.	SENG 265 Software Development Methods
Section(s)	A01 and A02
CRN	20422, 20423
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: Dynamic memory in C:

a. Must only be used for arrays.
b. Is unavailable for use with program- and file-scope variables.
c. Involves the use of "malloc()" and "free()".
d. Involves the use of "new()" and "free()".
e. None of the above.

Question 2: In the C language, "struct"s:

a. Are aggregate data types where each field may have a different type.
b. Are aggregate data types where each field must have the same type.
c. May contain fields that are pointers.
d. Associate functionality with a data type just like "class" does in an OO language (Java or C#).
e. None of the above.

Question 3: A function pointer in C:

a. Is a function that must take pointers as arguments.
b. Is a function that must return a pointer.
c. Is a function that must use pointers within the body of the function.
d. Is a function that must use dynamic memory.
e. None of the above.

Question 4: Every C source-code file:

a. Must contain the "`main()`" function as otherwise the file cannot be part of a C program.
b. Must `#include` a header file.
c. Must include function prototypes for all other functions in all other modules.
d. Can have variables that are only visible within the file and not outside of it.
e. None of the above.

Question 5: The C string function "`strncpy(foo, bar, baz)`":

a. Takes the first `baz` characters in "`bar`" and copies it to "`foo`".
b. Takes the first `baz` characters in "`foo`" and copies it to "`bar`".
c. Automatically allocates memory for the destination string.
d. Is considered by some safer to use than "`strcpy`".
e. None of the above.

Question 6: The `#define` preprocessor directive in C:

a. Must be used to define new data types in a C program.
b. Must be used to define new function prototypes in a C program.
c. Can be combined with `#ifndef` and `#endif` to implement conditional compilation.
d. Must be combined with `#ifdef` and `#endif` to create constants.
e. None of the above.

Question 7: A Python *list comprehension*:

a. Is the set of functions that Python permits to be called on lists.
b. May make use of a filtering expression
c. Always produces a new (possibly empty) list.
d. Converts list from a mutable form to an immutable form.
e. None of the above.

Question 8: Parameters in a Python function declaration:

a. May be annotated with a type.
b. May have default values.
c. May have a name.
d. May not refer to a Python function.
e. None of the above.

Question 9: Python modules:

a. Sometimes contain classes and functions.
b. Are "#included" into other Python programs.
c. Must be used to access the "print" function.
d. Are needed if object-orientation is to be used in a Python program.
e. None of the above.

Question 10: 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 11. To create a C-like switch statement in Python:

a. we should use the `case` statement.
b. we should use the `switch` statement
c. we should branch to a nested C script.
d. can use map on a list.
e. none of the above.

Question 12: When writing the body of a function in Python:

a. You must always refer to the `self` variable.
b. The block of the function definition must be indented.
c. You must end the function with the `return` keyword.
d. Your code may call other Python functions.
e. None of the above.

Question 13: 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 14: 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 15: After executing "`svn add blarg.c`", the user who issued this command:

a. Knows the file named "`blarg.c`" is part of the working copy tracked by Subversion.
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 16: 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 17: The `unittest` framework in Python:

a. can be used to associate test cases with class files we write in Python.
b. requires special scripting support from Unix to run the tests.
c. has methods for setting up and tearing down data structures needed for tests.
d. automatically executes the tests we have written.
e. none of the above.

Question 18: Regular expressions in Python:

a. Are used to specify sets of strings.
b. Are accessible via either `re.match` or `re.search`.
c. Can be compiled in order to reduce the time needed to perform a match.
d. Are sometimes used as part of control-flow expressions.
e. None of the above.

Question 19: In the context of this course's treatment of software-development process, a development lifecycle:

a. Has a "requirements" phase.
b. Has a "customer negotiation" phase.
c. Has a "library assessment" phase.
d. Has a "system design" phase.
e. None of the above.

Question 20: Dependencies in a "makefile" rule:

a. Consist of a list of commands.
b. Consist of a list of files.
c. Consist of a list of targets.
d. Consist of a list of base rules.
e. None of the above.

Section B: Python (60 marks)

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

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

def main():
    (aaa, bbb, ccc) = (1, 0, {})

    ddd = sys.stdin.readlines()

    while (ddd != []):
        eee = ddd[0]
        fff = eee.strip().split()
        for ggg in fff:
            if ggg in ccc.keys():
                ccc[ggg] = ccc[ggg] + ",. " + ("%d" % aaa)
            else:
                ccc[ggg] = ("%d" % aaa)
        aaa = aaa + 1
        ddd = ddd[1:]

    hhh = ccc.keys()
    print hhh
    hhh.sort()
    print hhh

    hhh = hhh[0:5]

    for h in hhh:
        print h, ":", ccc[h]

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

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

```
let us not
to the marriage
of true minds admit impediments.
```

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:

20130312	8.73	9.78	8.43	9.57	355002000	9.44
20130313	9.9	9.98	9.3	9.62	216287500	9.49
20130311	9.23	9.36	8.31	8.49	251329800	8.37
20130316	9.97	10.36	9.6	9.66	226193300	9.52
20130310	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", "20130311", "20130313")
```

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.

(This next page is available, if needed, for your answer to Question 22.)

(This page is available, if needed, for your answer to Question 22.)

```
input = open(filename, 'r')
LC = []  % line content
for line in input:
    LC = line.split()
    if int(LC[0]) >= start and int(LC[0]) <= high:
        days += 1
        totalvol += int(LC[5])
        if float(LC[2]) > highest:
            highest = float(LC[2])
        if float(LC[3]) < lowest:
            lowest = float(LC[3])

avg_vol = total_vol / days
output = (highest, lowest, avg_vol)
return output
```

Question 23: Weight 20

(a) What is value of `list2` produced below given the following code? Show all work.

```
def functionA (m, n):
      return (m * 10) - n;

list1 = [(12, 10, 5), (5, 2, -22), (5, 5, 1)]
list2 = [functionA(r, s) for (r, t, s) in list1]
```

120-5 50-(-22) 50-1

list2 = [115, 72, 49]

(b) Consider the following code for a "student" class:

```
class Student:
    """A class representing a student."""
    def __init__(self, n, a):
        self.full_name = n
        self.age = a
    def get_age(self):
        return self.age
```

Write the code needed for a subclass of "Student" called "Highschool_Student" where there is an additional field called "homeroom_teacher", an additional method called "get_homeroom_teacher", and a suitable constructor.

```
class Highschool_Student(student):
    def __init__(self, n, a, h):
        student.__init__(n, a)
        self.homeroom_teacher = h

    def get_homeroom_teacher(self)
        return self.homeroom_teacher
```

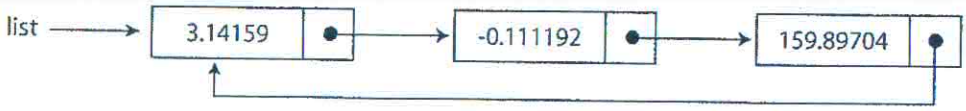
Section C: C programming (50 marks)

Question 24: Weight 20

Consider the following data structure:

```
typedef struct Node_t Node_t;
struct Node_t {
    float value;
    Node_t *next;
};
```

Also consider the following diagram showing one possible configuration of nodes using the data structure defined above:



Assuming the data-structure declaration in the box above is already given in of some C file, *write the remaining C code necessary in order to create dynamically* the structure shown in the diagram. Some marks will be given for the quality of your answer.

(The next page is available, if needed, for your answer to Question 24.)

(This page is available, if needed, for your answer to Question 24.)

Question 25: Weight 30

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 allocated and 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)
int *b= calloc( 1, sizeof(*a) );
int j=0;
int i=0;
while ( a[i] >=0 ){
    if ( a[i] > 0){
        b[j] = a[i];
        j++;
    }
    i++;
}
b[j]= a[i]; // copy the last element

return b;
```

Section D: Software-Development Process (20 marks)

Question 26: Weight 20

(a) Provide the names of at least four phases in software-development processes, and briefly describe the outcome of each phase.

requirements - meetings take place to figure out what the software should do (requirements)

design - building the architecture both hardware and software

implementation - work is divided into modules/units and actual coding is started

verification - functional testing such as unit testing is done

deployment - software is deployed

maintenance - when bugs come up they are solved

(b) Why is the "waterfall" process model not ideal? Explain how some other process model addresses one of the waterfall model's deficiencies.

waterfall is not ideal because it cannot easily adapt to changing requirements. Agile is much more iterative than waterfall making it easier to adapt to new requirements

Section E: Make (10 marks)

Question 27: Weight 10

Consider the following makefile:

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

edit : main.o kbd.o command.o display.o insert.o search.o files.o utils.o
        gcc -o edit main.o kbd.o command.o display.o insert.o \
        search.o files.o utils.o

main.o : main.c defs.h
        gcc $(FLAGS) -c main.c
kbd.o : kbd.c defs.h command.h
        gcc $(FLAGS) -c kbd.c
command.o : command.c defs.h command.h
        gcc $(FLAGS) -c command.c
display.o : display.c defs.h buffer.h
        gcc $(FLAGS) -c display.c
insert.o : insert.c defs.h buffer.h
        gcc $(FLAGS) -c insert.c
search.o : search.c defs.h buffer.h
        gcc $(FLAGS) -c search.c
files.o : files.c defs.h buffer.h command.h
        gcc $(FLAGS) -c files.c
utils.o : utils.c defs.h
        gcc $(FLAGS) -c utils.c
clean:
        rm -f main.o kbd.o command.o display.o insert.o search.o \
           files.o utils.o edit
```

(a) What rules must be executed if `command.h` is modified? Refer to rules by their target.

(b) What rules must be executed if `defs.h` is modified? Refer to rules by their target.

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

(d) How many rules does this makefile contain?

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

END

$$\frac{dT}{dt} = \alpha(T_H - T_C)$$

This page is for the sole use of examination evaluators.

Section A	/40
Section B	/60
Section C	/50
Section D	/20
Section E	/10
Total	**/180**

$$\begin{bmatrix} \sigma_x - \sigma_1 & \tilde{\tau}_{xy} & \tilde{\tau}_{xz} \\ & \sigma_y - \sigma_1 & \tau_{yz} \\ & & \sigma_z - \sigma_1 \end{bmatrix} \begin{bmatrix} l_1 \\ m_1 \\ n_1 \end{bmatrix}$$

$$l_1^2 + m_1^2 + n_1^2 = 1$$

$$(\sigma_x - \sigma_1) l_1 + \tilde{\tau}_{xy} m_1 + \tilde{\tau}_{xz} n_1 = 0$$