University
of Victoria

University of Victoria
Final Examination
April 2015

Course Name & No.	SENG 265 Software Development Methods
Section(s)	A01
CRN	22680 (A01) 22681 (A02)
Instructor	Michael Zastre
Duration	Three (3) hours

Name:
Student Number: V00 ____________

This exam has a total of 17 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 200.

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: Arrays in C:

a. May be safely extended by writing beyond the end of the array.
b. Always contain within them a field storing the size of the array (i.e., number of elements).
c. Are sliced using the colon operator (e.g., `sum[1:9]`).
d. Can be declared, amongst other places, in file scope or program scope.
e. None of the above.

Question 2: A `for` -loop in C:

a. Can contain other loops within it as nested loops.
b. Is a bottom-tested loop just like the `do-while` statement.
c. Must not contain statements that dereference pointers.
d. May refer to two variables in its loop-condition expression.
e. None of the above.

Question 3: When using `malloc()`, `calloc()` or `realloc()`:

a. We are always guaranteed to obtain from the heap the amount of memory we request.
b. A wrapper function can be written that takes care of examining return values for `NULL`.
c. It is good practice to cast the result to some pointer type.
d. Code should call `free()` once memory obtained from these calls is no longer needed (i.e., in order to prevent memory leaks).
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. May `#include` a header file.
c. Must include function prototypes for all other functions in all other modules.
d. Can have variables only visible within the file and not outside of it.
e. None of the above.

Question 5: The C string function `strtok(somestring, someseparatorchars)`:

a. Must be followed at some point with `strtok(NULL, someseparatorchars)` in order to continue tokenizing the original string.
b. Can only be used when we `#include <strtok.h>`.
c. Automatically allocates memory for all tokens extracted from the string.
d. Returns the value `-1` when there are no more tokens in the original string.
e. None of the above.

Question 6: Python variables:

a. May refer to functions.
b. Can be assigned values of different types within the same function.
c. Can never be global as global variables must be statically typed in all languages.
d. May be assigned the value of a tuple.
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: Python code blocks:

a. Can be used to represent the true- and false-code of an `if-else` statement.
b. May contain other Python blocks.
c. Are indicated by indentation.
d. Are the only structures in which mutable tuples may be constructed.
e. None of the above.

Question 9: When using Python lists:

a. Slices of a list may be accessed with syntax such as `somelist[start:end]`.
b. A negative lookup such as `somelist[-3]` refers to list values positioned relative to the end of the list).
c. A copy of list contents is made as the result of a statement such as `some_other_list = somelist`.
d. We can refer to the list from the second element to the end using `somelist[:2]`.
e. None of the above.

Question 10: To create a C-like switch statement in Python, we:

a. Should use the `case` statement.
b. Should use the `switch` statement
c. Should branch to a nested C script.
d. Can use `map` on a list.
e. None of the above.

Question 11. Considering the Python expression `(True and (x > 5) and bar)`:

a. It generates a syntax error as `bar` always represents a function and must be called with parentheses.
b. Its value may be `True` or `False`.
c. Its value may be `False` or the value of `bar`.
d. It may be used to control the iterations of a `while` loop.
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: After we use `svn add` on a file in our project:

a. Every change to that file is automatically sent to the Subversion repository.
b. Every additional file added to that directory is automatically sent to the Subversion repository.
c. All other developers who have a copy of the project must perform the run `svn add` command.
d. A new file will be stored in the directory that is the concatenation of all changes made to the file.
e. None of the above.

Question 14: The UNIX "`rm`" command:

a. Can be used to delete text files.
b. Can be used to delete executable files.
c. Can be used to enlarge a text file (i.e., "more room").
d. Can be used to enlarge an executable file (i.e., "more room").
e. None of the above.

Question 15: Again considering `svn add`, after executing "`svn add categorise.c`" within an existing Subversion project, the user who issued this command:

a. Knows the file named "`categorise.c`" is part of the working copy tracked by Subversion.
b. Knows the file named "`categorise.c`" has been committed to the repository.
c. Has created a new file in their working module.
d. Has taken one of a series of steps that may allow another user to access the contents of "`categorise.c`".
e. None of the above.

Question 16: `svn checkout http://district12.pn/everdeenk`

a. Can only be performed once for "`everdeenk`".
b. Will return an error if some other user has already checked out "`everdeenk`".
c. Creates possibly many subdirectories.
d. Is the opposite of "`svn checkin http://district12.pn/everdeenk`".
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. May be used to identify and extract portions of strings.
b. Are only used to match strings and never for substitution into strings.
c. Make heavy use of meta-symbols.
d. Use the symbol `\d{+num}` to indicate the distance (here equal to `num`) in characters from the start of the searched string to an occurrence of the pattern itself.
e. None of the above.

Question 19: Dependencies in a "Makefile" rule may:

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 variables.
e. None of the above.

Question 20: In the Knuth-Barr categories of bugs:

a. The "algorithm" category means the code as written followed the intent of the programmer but the intent was wrong.
b. The "forgotten" category means the code reads or writes incorrect data or accesses a wrong storage location.
c. The "distracted" category means the code was written by overcommitted and exhausted developers who introduced bugs as a result of deadline pressures.
d. The "compiler" category means the bug was introduced into the source code by an incorrectly written compiler.
e. None of the above.

(This page intentionally left blank.)

Section B: Python (Total marks: 70)

Question 21 (20 marks): Consider the following Python code, named "`mystery.py`" (line numbers provided for your reference):

```
 1 #!/usr/bin/python3
 2
 3 import sys
 4
 5 def main():
 6     aaa = []
 7     for bbb in sys.stdin:
 8         ccc = bbb.split()
 9         aaa.extend(ccc)
10
11     ddd = [len(zzz) for zzz in aaa]
12     ddd.sort()
13
14     eee = ddd[0]
15     fff = 1
16     for ggg in ddd[1:]:
17         if eee != ggg:
18             print (eee, ":", fff)
19             eee = ggg
20             fff = 1
21         else:
22             fff = fff + 1
23
24     print (eee, ":", fff)
25
26 if __name__ == "__main__":
27     main()
```

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

```
Do you like green eggs and ham?
I do not like them Sam I am.
I do not like green eggs and ham.
```

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`"). Assume the file permissions for `mystery.py` are correct and the Python interpreter is located as shown in the bang path. *You may use page 9 for your rough work.*

Question 22. Weight 50

A text file contains a sequence of x & y coordinates. These represent the endpoints of line segments (i.e., in example 1, the first line segment is from [0, 0] to [0, 5], the next segment is from [0, 5] to [5, 5], etc.).

Example file 1:

```
    0 0
    0 5
    5 5
    5 0
    0 0
```

Example file 2:

```
    3 5
    10 3
    2 1
```

Every such file contains information for at least one line segment. You may not assume a maximum number of such line segments.

a) Write a Python function `computeDist coord1 coord2` to compute and return the straight-line distance between a pair of coordinates [x1, y1], [x2, y2]:

$$dist = \sqrt{(y1 - y2)^2 + (x1 - x2)^2}$$

The exponentiation operator in Python denoted by "**". You can obtain the square root of a number by taking it to the power of 0.5. (Some marks will be given for the quality of your answer.)

b) Write a Python script that uses the function from (a) to print sum of the distances of all segments in a text file. Assume all input is from stdin. The value printed for example 1 would be 20; that for example 2 would be 15.526. (Some marks will be given for the quality of your answer.)

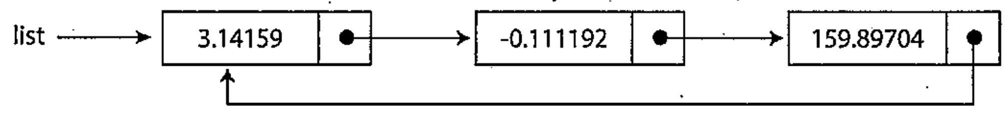
Section C: C programming (Total marks: 70)

Question 23: (25 marks)

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* with dynamic memory the structure shown in the diagram. Some marks will be given for the quality of your answer.

The next page (page 13) is available for your work / answer if needed.

Question 24: (45 marks)

Write a C function to determine if one string is an **anagram** of another. Your function *is_anagram()* will take two strings as parameters, and will returns 1 if the strings are anagrams of each other, otherwise it will returns 0.

An anagram is a phrase that uses all the letters—and exactly those letters—of some other phrase. Below are some anagram examples:

- Zastre: *Ersatz*
- Stephen Harper: *Eh! Rent, perhaps?*
- Dormitory: *Dirty Room*
- The check is in the mail!: *Claim "Heck, I sent it (heh)"*
- I sat there with Sally: *Aha! Lithely twisters!*

Notice the number of spaces and use of punctuation in an anagram can differ greatly from the original phrase. *We ignore spaces and punctuation when checking if two phrases are anagrams of each other.*

As a guide towards a solution, assume you can use an already-written function named *alpha_to_index* which takes a single parameter (a character). It returns 0 if the character is not an alphabetical character, otherwise it returns a value from 1 to 26 if the character is a letter from the English alphabet. (When the function is passed 'A' or 'a' as the argument, it returns 1; when 'B' or 'b' is passed, it returns 2; etc.)

The next page (page 15) is available for your work / answer if needed.

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

Section D: Debugging (20 marks)

Question 25. Weight 20

Recalling your experiences writing solutions to the course assignments, and in the context of material on debugging covered in this course, answer the following questions.

(a) Give an example of a program *failure*, and an example of a program *error*.

(b) Explain how the examples relate to a program *error*, and given an example of such an error.

(c) Referring to the categories described in lectures, name a subcategory of *algorithm bugs*, and give an example.

(d) Briefly describe two bug-*avoidance strategies*.

END OF EXAM

This page is for the sole use of examination evaluators.

Section A	/40
Section B	/70
Section C	/70
Section D	/20
Total	**/200**