University of Victoria

University of Victoria Final Examination August 2017

Course Name & No.	SENG 265 Software Development Methods
Section(s)	A01
CRN	30721 (A01)
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.

All cellphones must be 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 (60 marks): 20 questions

For each question in this section, place an X beside <u>all</u> answers that apply. Each question is worth three (3) marks. *Partial marks are <u>not</u> given for incomplete answers.*

Question 1: The boolean `false` value in C:

___ is represented by the C string "false".

___ is returned by `malloc()` when this function succeeds in obtaining heap memory.

___ is used by C to determine the bounds of a character array.

___ may have its C equivalent stored in an integer variable.

___ None of the above.

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

___ may be used to traverse the nodes of a linked-list implementation as discussed in this course.

___ uses a very different syntax compared to Python's `for` statement.

___ is limited to at most two per function.

___ may refer to more than one variable in its loop-condition expression.

___ None of the above.

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

___ we are always guaranteed to obtain from the stack the amount of memory we request.

___ we are always guaranteed to obtain from the heap the amount of memory we request.

___ we must always specify the amount of memory requested, possibly in an expression involving the `sizeof()` macro.

___ the memory allocated in one module cannot ever be accessed by code in another module.

___ None of the above.

Question 4: Every C source-code file:

___ must contain the `main()` function as otherwise the file cannot be part of a C program.

___ may *include* a header file.

___ may include function prototypes for all other functions in all other `.c` files.

___ can run Python code by including such code in a `#define`.

___ None of the above.

Question 5: The C function `fwrite`:

___ opens the filename corresponding to the string passed as the first parameter.

___ takes an address to memory that is to be written to the file.

___ takes a number indicating the quantity of items to be written.

___ may only be used with the `main()` function.

___ None of the above.

Question 6: A Python 3 variable:

___ can have a type in the same way variables are declared in C.

___ can be assigned values of different types at different points within the same function.

___ may be assigned the value of a list.

___ may be assigned the value of a tuple.

___ None of the above.

Question 7: Assuming `list1 = [“the”, “quick”, “brown”, “fox”]` then after the following loop:

```
list2 = []
for t in list1:
    if len(t) < 4:
        list2.append(t)
```

it is the case that *list2* is equal to:

___ `[“the”, “the”, “quick”, “quick”, “brown”, “brown”, “fox”, “fox”]`

___ `[“(the, the”), “(fox, fox)”]`

___ `[(“fox”, “fox”)]`

___ `[(“the”, “the”)]`

___ None of the above.

Question 8: When defining a Python 3 class:

___ methods have `self` listed as the first parameter.

___ we are able to write constructors as needed.

___ a class variable’s name may match an instance variable’s name.

___ that class may be contained within a file having a name different than the class’s name itself.

___ None of the above.

Question 9: When using Python 3 lists ______.

___ a negative lookup (such as `somelist[-3]`) refers to list values positioned relative to the end of the list

___ a copy of the list contents is made as the result of a statement such as `some_other_list = somelist`

___ we can refer to the list from the second element to the end using `somelist[1:]`

___ Python 3 ensures all values within the same list have the same type

___ None of the above.

Question 10: A Python 3 function:

___ may be passed as a function parameter.

___ may or may not have parameters.

___ are forbidden from containing underscores (i.e., "_") as part of the function name.

___ may be returned as a result of some function's execution.

___ None of the above.

Question 11. Considering the Python 3 expression `(False or (y < 10) or bar)`:

___ it generates a syntax error as *bar* always represents a function and must be called with parentheses.

___ is equivalent to the Python 3 expression `((y < 10) or bar)`.

___ its value may be `True` or the value of `bar`.

___ its value may be passed as a parameter to some function.

___ None of the above.

Question 12: Consider the following Python 3 dictionary:

```
usprez = {42:'clinton', 41:'ghwbush', 43:'gwbush', 44:'obama'}
```

Place an *X* beside each line below that ***does not result in an error*** when evaluated by a Python 3 interpreter.

___ `print(usprez[42])`

___ `len(usprez[41])`

___ `usprez[40] = "rreagan"`

___ `output = "President " + usprez[45]`

Question 13: `git clone ssh://jtrudeau@git.seng.uvic.ca/seng265/jtrudeau`:

___ Is accessing what is usually referred to as a "remote repository".

___ Will return an error if some other user has already performed a `git push` on the `jtrudeau` repository.

___ Creates possibly many subdirectories.

___ Includes the full commit history of the project stored at that remote repository.

___ None of the above.

Question 14: The git `push` command:

___ automatically commits changes to files in the current local repository.

___ is the same as the `clone` command.

___ normally results in the most recent commit (amongst others) being transferred to some remote repository.

___ reverses the effect of the most recent `pull` command.

___ None of the above.

Question 15: The UNIX `diff` command:

___ is part of the math library which provides a differentiation utility for symbolic math.

___ automatically tests files for completion.

___ reports the ways in which two files do not have the same content.

___ may only ever be used as part of a piped Unix command.

___ None of the above.

Question 16: Consider the following regular expression:

```
"^\w(tt|t)\w+$"
```

Place an X beside each string below that is found by the regex (and note that the quotation marks are only used to denote the start and end of the string). Assume the Python 3 *re* module is being used, that the *search* function is provided only two parameters (i.e., the regex plus the string to be checked against the regex), and that raw strings are always used for a regular expression.

___ `"tantalizing"`

___ `"attention"`

___ `"cattle"`

___ `"often"`

___ `"otiose"`

Question 17: Consider the following string:

`"604-555-1515"`.

Place an X beside each regular expression below for which a match is found with the string. (The same assumptions about Python 3 given in question 15 apply here.)

__ `"\d{3}-\d{3}"`

__ `"^\d{3}-\d{3}$"`

__ `"(4|5|6).(4|5|6)"`

__ `"^.*[1-5].*$"`

__ `"^.*[1-5]-[1-5]$"`

Question 18: Place an X beside each regular expression that matches the string `"in"` and the string `"into"` but not `"bin"` or `"cinch"`. (The same assumptions about Python 3 given in question 15 apply here.)

__ `"\bin\b"`

__ `"\bin"`

__ `"in\b"`

__ `"^ch\b"`

__ `"\bin$"`

Question 19: The `git checkout` command:

__ is another way of cloning a remote repository.

__ requires special scripting support from Unix to run the tests.

__ creates a new branch within the remote repository.

__ is used to switch the `git` working directory to some other branch (if the branch exists).

__ none of the above.

Question 20: A *merge conflict* in `git`:

__ can be resolved in part by editing files involved in the conflict.

__ always happens when `git merge` is performed.

__ occurs when the program executable cannot be successfully built using make.

__ can only be handled by cloning a new copy of the repository.

__ None of the above.

Section B: Python 3 (Total marks: 50)

Question 21 (15 marks): Consider the following Python 3 code, named `mystery.py` (line numbers provided for your reference):

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

and the following input in a file named `eecummings.txt`:

```
a little whos
(he and she)
under are this
wonderful tree
```

Write on the lines below the output that results when the input is redirected into the Python program (i.e., `cat eecummings | ./mystery.py`). Assume the file permissions for `mystery.py` to execute as a command are correctly set and the Python 3 interpreter is available given the bang path. *You may use the next page for your rough work.*

Question 22: (15 marks)

Write a Python 3 function to determine the number of vowels (i.e., "a", "e", "i", "o" and "u") are in a string passed to the function. Name the function `num_vowels`.

Question 23: (10 marks)

Write a Python 3 list comprehension taking a list of strings named *li* that produces a new list of all those strings with less than three vowels. For example, if `li` is `["the", "antelope", "and", "the", "aardvark", "fell", "in", "love"]` then the result must be `["the", "and", "the", "fell", "in", "love"]`. You may assume `num_vowels()` from question 22 is correctly implemented.

Question 24: (20 marks)

Write a Python 3 function called that accepts a string as input and returns *True* if the string is a palindrome, otherwise it returns `False`. A *palindrome* is a string spelling the "same" forwards and backwards.

Here are some examples of palindromes:

- *radar*
- *abaaba*
- *anna*
- *hanah*
- *was it a rat i saw*
- *yo, banana boy*

True palindromes usually include all forms of punctuation and a mix of upper- and lower-case characters. However, for this function you may assume the only punctuation is either a single space between words or a comma. You may also assume all characters are in lower case.

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

Section C: C programming (Total marks: 60)

Question 25 [30 marks]

Write a C function `int undo_runs(char *line)` that accepts a C string which modifies the array `line` such that all runs of characters are removed, with the number of characters removed being returned by the function. For example:

Here are some examples (although note that your solution must work also work for characters other than those shown in the examples):

- `undo_runs("aaaaabbb")` modifies the contents of the array to "`ab`" and returns 6
- `undo_runs("abababab")` does not modify the contents of the array and returns 0
- `undo_runs("aaabbbaaabbb")` modifies the contents of the array to "`abab`" and returns 8
- `undo_runs("aaaaababb")` modifies the contents of the array to "`abab`" and returns 5
- `undo_runs("teetotaller")` modifies the contents of the array to "`tetotaler`" and returns 2
- `undo_runs("")` does not modify the contents of the array and returns 0
- `undo_runs(NULL)` returns 0

Note that the array passed to the function *must be modified in place.* For example, given the following code:

```
char s[20];
strncpy(s, “aaaaababb”, 20);
int result = undo_runs(s);
printf(“%d %s\n”, result, s);
```

the output produced would be:

```
4 abab
```

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

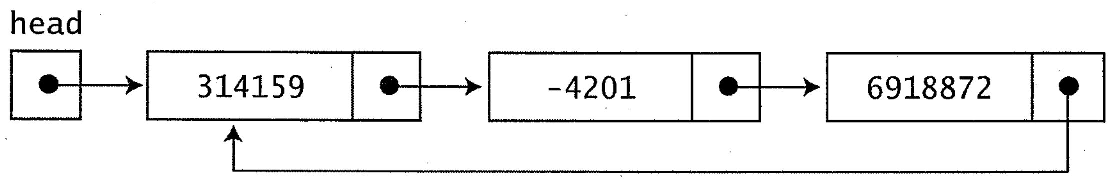
(The next blank page of this exam may be used for your answer.)

Question 26: (30 marks)

Consider the following C code:

```
typedef struct list_node_t list_node_t;
struct list_node_t {
    int          value;
    list_node_t *next;
};
```

Also consider the following diagram showing one possible configuration of nodes using the type declaration above:



Assuming the data-structure declaration in the box above is already given in some C file, write the remaining C code necessary in order to create – using `malloc()` – the linked-list structure shown in the diagram.

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

(The next blank page of this exam may be used for your answer.)

END OF EXAM

This page is for the sole use of examination evaluators.

Section A	/60
Section B	/60
Section C	/60
Total	**/180**