SENG 265 – Software Development Methods
Fall 2023
Midterm Exam: Wednesday, 18 October 2023
Instructor: Roberto A. Bittencourt

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

- **All answers are to be written on this exam paper.**
- We will not answer questions during the exam. If you feel there is an error or ambiguity, write your assumption and answer the question based on that assumption.
- The exam is closed book. No books or notes are permitted.
- When answering questions, please do not detach any exam pages!
- ***Electronic devices are not permitted***. Cellphones must be turned off.
- The marks assigned to each section are printed within parentheses. Partial marks are available for questions in Sections B and C.
- There are ten (10) printed pages in this document, including this cover page.
- Last page is left blank for scratch work. **Answers on the that page will not be graded.**
- We strongly recommend you read the entire exam through from beginning to end before starting on your answers.
- **Please have your UVic ID card available on the desk for inspection.**

Section A (30 marks +10 extra marks): For each question in this section, place an X **beside all answers that apply.** Each question **is worth 4 (four)** marks. ***Partial marks are given for incomplete answers.***
GRADING RUBRIC:
FOR EVERY CORRECT ANSWER MARKED, 0.8 marks
FOR EVERY INCORRECT ANSWER NOT MARKED: 0.8 marks

Question 1: By stating that *git* is a *DISTRIBUTED* version-control system, we are saying:

__X__ various repositories may be used by clients to deal with a single software project.

____ if the remote repository breaks down, it is not possible to recover commit history.

__X__ each user with access to a remote repository may keep a copy of the repository history at its own computer.

__X__ It permits concurrent read and write access of remote repositories to users who have read- and write-access rights to those repositories.

____ None of the above.

Question 2: *git clone ssh://billg@git.seng.uvic.ca/seng265/billg*

__X__ copies a full remote repository to a local repository.

____ gives the local repository access to only the latest commit made to that remote repository.

____ Can only be performed once for the *billg* project.

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

____ None of the above.

Question 3: The *git commit* command:

____ is the same as the git *push* command.

____ sends all the staged files in the staging area to a remote repository.

____ sends all the staged files in the staging area to the remote central server.

____ reverses the effect of the most recent *git clone* command.

__X__ None of the above.

Question 4: If a directory has the permissions *drwxr-xr--,* then this means:

____ anyone with access to this system is able to enter the directory.

__X__ anyone with access to this system is able to list the directory contents.

____ the owner of the directory cannot remove the execute permission of the group the directory belongs to.

__X__ only the owner is able to delete the directory.

____ None of the above.

Question 5: The UNIX *mv* command:

____ Can be used to create symbolic links to an existing file as a type of "shortcut".

__X__ Can be used to rename a file.

__X__ Can be used to move a file to another directory.

____ Can be used to delete directories if used with the –r recursive option.

____ None of the above.

Question 6: A pipe ("|") that may be constructed using the bash shell:

__X__ connects the *stdout* stream of one command with the *stdin* stream of the next command.

____ changes all the file streams in the second command in the pipe to the *stdout* stream.

____ connects the *stdout* stream of one command with the *stdout* stream of the next command.

____ redirects the standard output of the command after the pipe to a file.

____ None of the above.

Question 7: The *while*-loop in C:

_____ Is a bottom-tested loop just like the *for* statement.

__X__ May contain assignment commands inside its parentheses (round brackets).

__X__ May contain statements that dereference pointers.

_____ Will execute the commands inside its body at least once.

_____ None of the above.

Question 8: The C string function *strncpy(char *dest, char *src, int len):*

_____ Copies at most *len* characters from *dest* to *src*.

_____ Always copies exactly *len* characters from *src* to *dest*.

_____ Is considered less safe to use than *strcpy*.

_____ Raises a *StringLengthException* if *src* string is longer than *dest* string.

__X__ None of the above.

Question 9: When using Python 3 lists, _____ .

_X_ a copy of the list contents is made as the result of a statement such as `some_other_list = somelist[:]`

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

_X_ given some list, we can refer all the values from the second item to the fifth item using `somelist[1:5]`

___ immutable types such as tuples cannot be added as list elements

___ None of the above.

Question 10: A Python 3 function:

___ always declares the type of each its parameters when there are parameters.

_X_ may be passed as a function parameter.

___ may be overloaded, i.e., may allow other functions with the same name to be defined with a different number of parameters and a different body.

_X_ always has a return value, even if no return statement is called inside the function.

___ None of the above.

Section B (30 marks + 10 extra marks): Two questions, each one worth 15 (fifteen) marks.

Question 11: (20 marks). Consider the following Python 3 code:

```
 1 def f(q, w):
 2     return q(w)
 3
 4 def f2(n):
 5     for i in range(n):
 6         # using end='' does not add a newline at print end
 7         print('%d ' % i , end='')
 8     print()
 9
10 def f1(x):
11     for i in range(1, x + 1):
12         f(f2, i)
13
14 def main():
15     number = 5
16     f(f1, number)
17
18 main()
```

Trace the code's output below, as it would appear in the console screen.

_____0 _____

_____0 1 _____

_____0 1 2 _____

_____0 1 2 3_____

_____0 1 2 3 4 _____

Function f is an applier function, which receives any function and a value as inputs and returns the function applied to the value.

Partial credit may be given **(10 points)** if it can be inferred that the student understood the question does a nested loop that counts one more element at each iteration of the outer loop, but got confused with the loop limits (which starts from number 0 and ends in number 4).

Question 12: ***(20 marks)*****.** Consider the following C program (with line numbers provided in the left-hand margin for your convenience):

```
 1 int main() {
 2     int *p, *q, *r, *s, *t, *u;
 4     int a=0, b=0, c=0, d=5, e=0, f=0;
 5
 5     p = &b;
 6     q = &d;
 7     s = &a;
 8     r = &c;
 8     *s = 10;
 9     *r = a + *q;
10     t = s;
11     *p = *r;
12     u = &c;
13     *u = (*p + *r) *  *s;
14     e = *q - b;
15     /* what is the state of the memory at this point
16     before the main function ends? */
17 }
```

Suppose that the image below represents the memory of the program above. Fill out the content of the memory cells before the main function ends (after line 14).

address	Memory	*variable*
10500	10528	**p**
10504	10536	**q**
10508	10532	**r**
10512	10524	**s**
10516	10524	**t**
10520	10532	**u**
10524	10	**a**
10528	15	**b**
10532	300	**c**
10536	5	**d**
10540	-10	**e**
10544	0	**f**

2 marks for each correct value in the memory cells, either the correct supposed address coming from the first column for the pointers or the correct integer value for the integer variables.

Section C (40 marks): One question

Question 13

Write a C function with the following signature:
`void remove_duplicate_characters(char *source, char *destination)`
This function accepts two strings `source` and `destination` as parameters and modifies the `destination` string so that it contains only the unique chars in `source`. The original `source` string must not be modified. *Memory for the resulting string must NOT be dynamically allocated*.
The characters in the `destination` string must be ordered in the same order they originally appear in the `destination` string. For example, if the following commands are run:

```
char source[100] = "house of the rising sun";
char destination[100];
remove_duplicate_characters(char *source, char *destination);
printf("%s\n", destination);
```

the console should show:

```
house ftring
```

Do <u>not</u> modify the contents of the `source` string passed to your function as a parameter.

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

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

There are various ways to solve this question. Using pointers or not to index a string, using a while and a for, using two for loops, using one loop and an auxiliary function, and so on.

When grading, consider (NEW RUBRIC):

- **5 marks: Outer loop correctly iterating over source string**
- **10 marks: Inner loop or function correctly iterating over partial destination string and checking previous elements of input string**
- **5 marks: adding the string terminator at the end of the destination string**
- **5 marks: Code quality (indentation, legibility)**
- **5 marks: Not hardcoding either input or output strings**
- **5 marks: gives a correct answer at the output**
- **5 marks: does not change the input string (strtok included)**

OPTION 1: USING POINTERS, A WHILE AND A FOR

```
void remove_duplicates(char *source, char *destination) {
    int i = 0;
    char *p = source;
    while(*p) {
        int found = 0;
        for (int j = 0; j < i; j++) {
            if (destination[j] == *p) {
                found = 1;
                break;
            }
        }
        if (!found) {
            destination[i] = *p;
            i++;
        }
        p++;
    }
    destination[i] = '\0';
}
```

OPTION 2: USING ARRAYS AND TWO FORs

```
void remove_duplicates(char *source, char *destination) {
    int k = 0;
    for (int i = 0; i < strlen(source); i++) {
        int found = 0;
        for (int j = 0; j < k; j++) {
            if (destination[j] == source[i]) {
                found = 1;
                break;
            }
        }
        if (!found) {
            destination[k] = source[i];
            k++;
        }
    }
    destination[k] = '\0';
}
```

(This page intentionally left blank for scratching.)