



## Midterm 2

Term: Fall 2018 (Sep4-Dec4)

### Student ID Information

Last name: \_\_\_\_\_ First name: \_\_\_\_\_

Student ID #: \_\_\_\_\_ CS.Dal.Ca userid: \_\_\_\_\_

Course ID:	<b>CSCI 2132</b>	Grade Table	
Course Title:	Software Development	Question	Score
Instructor:	Vlado Keselj	1	/10
Date of Exam:	8 Nov 2018	2	/12
Time Period:	Start: 19:35 End: 20:25	3	/12
Duration of Exam:	50 minutes	4	/8
Exam Location:		5	/10
Number of Exam Pages: 8 pages (including this cover sheet)		$\Sigma$	/52
Exam Type:	<b>Closed Book</b>		
Additional Materials Allowed:	<b>One letter-format paper (8.5"×11") with anything written or printed on it (both sides). No textbooks, computers, calculators, or other aids are allowed.</b>		

*Note: There are numerous questions in this exam that cover material we cover later. They are marked and will have no equivalent in Monday's exam.*

1. (10 points) **True-false questions:** 2 points each. Justification is not necessary, but brief justification may be helpful if correct.

a) (2 points) In the C programming language, the following two pairs of `scanf` format strings are equivalent: `"%c:%c"` and `"%c: %c"`

b) (2 points) The `-g` option of `gcc` is used to produce object code only from a C source file. 

c) (2 points) The following C code is valid: `int i; double d=5.1; i = d;`

d) (2 points) The function parameters and function local variables are stored on the call stack. 

e) (2 points) After executing `'int a[10]={1};'` the value of `a[10]` is not defined. 

2. (12 points) **Multiple-choice.** Circle the correct answer to the question.

a) (3 points) Which of the following statements is FALSE about processes?

- A. We can start a process in background by using character '&' in the command line.
  - B. A foreground process can print to the terminal.
  - C. A background process can read input from the keyboard.
  - D. A foreground process can run in the same time (concurrently) as a background process.
- 

b) (3 points) Which phase is NOT part of the Waterfall Model of software development life cycle? 

- A. Verification
  - B. Requirement Analysis
  - C. Prototype Development
  - D. Design
- 

c) (3 points) After the following code: 

```
int a[10]={10,20,30}; int *p; p=&a[2]; p -= 1; --(*--p);
```

the array a will start with the following values or an error is generated:

- A. {9,19,29}
  - B. {10,20,27}
  - C. Invalid pointer operation (possibly Segmentation-fault error)
  - D. {9,20,30}
- 

d) (3 points) The fork system call is used in the following situation:

- A. Creation of a new process.
- B. Creation of a new sub-directory.
- C. Execution of a conditional statement.
- D. Creation of a new stack frame.

**3. (12 points) Give concise answers.**



a) (4 points) Briefly describe gdb commands `break`, `step`, and `next`.

b) (4 points) If we have the declarations `int *p, a[10] = {1};` briefly explain the meaning of the statement: `p = a+2;`. Is there another way to write this statement?



c) (4 points) If we execute MergeSort on array `{4,1,3,7,6,2,5,8}`, how many times will the function `'merge'` be executed? What will be sub-arrays that are merged during the last execution of the `'merge'` function? (List the values of sub-arrays.)



**4. (8 points) Code snippets.**

a) (4 points) What is the output of the following code:

```
int a[] = {1,10,20}, *p=a, i=1, j=2;
for (i=0, p=a; i < 2; i++) {
    int j = *p;  p++;  *p = *p + j;
    printf("in: i=%d j=%d a=%d,%d,%d\n", i, j, a[0], a[1], a[2]);
}
printf("out: i=%d j=%d a=%d,%d,%d\n", i, j, a[0], a[1], a[2]);
```

b) (4 points) Write a C function `sort2` which can be used to do a “mini-sort” of two integer variables, by swapping their values only if the first variable is larger than the second. For example, after executing the following code: `int a=78, b=51; sort2(&a, &b);` the values of the variables would be `a=51 b=78`, but if we execute `sort2(&a, &b);` again, the values would not be changed.

**5. (10 points) C Program.**

We will call a sequence of integers a *slow-changing sequence* if difference between any two consecutive numbers in sequence is at most 1. Write a C program that reads a positive integer  $n$  and prints all slow-changing sequences of non-negative integers that start with 0 and have length  $n$ . For example, for  $n = 3$ , the program should print sequences: 0 0 0, 0 0 1, 0 1 0, 0 1 1, and 0 1 2. You do not need to check for errors in input.

(5 point option): For a partial solution of 5 points, write a function that checks whether an array is a slow-changing sequence.

(extra blank page 1 provided)

(extra blank page 2 provided)