Dalhousie University
CSCI 2132 — Software Development
Winter 2018
Midterm Examination II
March 12
15:37-16:24

Student Name:	_____
Student ID Number:	_____
FCS Username (CSID):	_____
Signature:	_____

Please Note: These solutions are for students in the Winter 2018 version of CSCI 2132 only. They may not be photocopied or distributed in any way, including electronically, to any other person without permission of the instructor.

Instructions (Read Carefully):

1. Aids allowed: one 8.5" by 11" piece of paper with anything written or printed on it (both sides). No textbooks, computers, calculators, or other aids.
2. This exam has 7 pages, including this page. Ensure that you have a complete paper.
3. Understanding the exam questions is part of the exam. Therefore, questions will **not** be interpreted. Proctors will confirm or deny any error or ambiguities only. If you are unsure of your own understanding, clearly state reasonable assumptions that will not trivialize the questions.
4. Marks will not be given for documentation, though feel free to add documentation if you feel necessary. You however should make your code readable to facilitate marking.
5. Questions are NOT sorted from the easiest to the most difficult.

1. [6 marks] True-false: 2 marks each. No justification necessary.

   (a) In the C programming language, function arguments are passed by value.

   true

   (b) In black box testing, we use internal knowledge of implementation to guide the selection of test cases.

   false

   (c) The stack frame of a C function stores external variables that are used by statements of this function.

   false

2. [6 marks] Multiple-choice – no justification necessary. Circle the *single* best answer. 3 marks each.

   (a) If `ch` is a variable of type `char`, which one of the following statements is illegal?

   A. `i += ch;    /* i has type int */`
   B. `ch++;`
   C. `ch = getchar();`
   Ⓓ. `printf(ch);`

   (b) Which of the following is the fourth of the five steps in the Waterfall Model that was proposed to describe software development life cycle?

   A. Implementation;
   Ⓑ. Verification;
   C. Requirements Analysis;
   D. Software Design.

3. [8 marks] Give concise answers to the following questions. 4 marks each.

   (a) Suppose that we have compiled a C program and generated an executable program named `binary` which is ready to be debugged using `gdb`. We have also entered `gdb binary` to debug it. Write down four `gdb` commands to perform the following tasks:

      i. Set a break point at the `main` function.

      ii. Run this program using gdb.

      iii. When the program pauses, run a `gdb` command such that each time we run one line of the program, the value of the variable `middle` in the program will be printed automatically by `gdb`.

      iv. Execute the next statement of the program. There are two commands that can perform this task. You are just required to give one of them and you are not required to state the difference between these two commands.

```
break main
run
display middle
step
```

   For the fourth command, you can also use `next` (make sure that you know the difference). You are allowed to use abbreviations of these commands.

   (b) For the following pair of `scanf` format strings, indicate whether or not the two strings are equivalent: `"%d-%d-%d"` versus `"%d -%d -%d"`

   If they are, simply say yes and no justification is required. If not, show how they can be distinguished by providing one input. You need not explain why this input distinguishes them.

   No

```
1 -1-1
```

4. [8 marks] Show the output produced by each of the following program fragments. You need not justify your answers.

   4 marks each.

   (a)
```
int i = 1;
int j = 2;

if (i) {
  int i;
  i = j;
  j = 0;
}

printf("%d %d\n", i, j);
```
   Solution:
```
1 0
```

   (b)
```
#include <stdio.h>

int mystery(int n);

int main() {
  int n = 3;

  printf("%d %d\n", n <= 1, mystery(n));

  return 0;
}

int mystery(int n) {
  if (n <= 1)
    return 1;
  else
    return (mystery(n-1) + n);
}
```
   Solution:
```
0 6
```

5. [11 marks] A palindromic number is a number that remains the same when its digits are reversed. For example, 14941 and 2332 are palindromic. 27671 is not: when reversing its digits, we get the number 17672 which is different from the original number.

   The following C program asks a user to enter a positive integer, and then tells whether this number is palindromic. Part of the program is given, and you are required to complete the program by filling in the missing code. There are two sub-problems:

   (a) In the `main` function, write the code that computes the digits of `num`, and store these digits in array `digits`. The variable `num_digits` will store the number of digits of `num`. For example, if `num` is assigned 14941, then after the execution of your code, `num_digits` will be 5 and the first five elements of `digits` will be 1, 4, 9, 4 and 1. The remaining elements are not used.

   (b) Implement the function `palindromic` (on the next page) as a **recursive function** that will check whether the digits `digits[lower]`, `digits[lower+1]`, ..., `digits[upper]` form a palindromic number, so that the function call in the `main` function (next page) would correctly complete the program.

   Error handling is not required. Two marks will be deducted if your implementation of the function `palindromic` is not recursive. You are allowed to define new variables in `main`, though there exists a correct solution that does not require you to do so.

```
#include <stdio.h>

#define LEN 30

int palindromic(int digits[], int lower, int upper);

int main() {
  int num;
  int digits[LEN];
  int num_digits = 0;

  printf("Enter a positive number: ");
  scanf("%d", &num);

  while (num != 0) {
    digits[num_digits++] = num % 10;
    num /= 10;
  }

  if (palindromic(digits, 0, num_digits-1))
    printf("It is a palindromic number.\n");
  else
```

```
    printf("It is not a palindromic number.\n");

  return 0;
}

int palindromic(int digits[], int lower, int upper) {
  if (lower >= upper)
    return 1;

  if (digits[lower] == digits[upper])
    return palindromic(digits, lower+1, upper-1);
  else
    return 0;
}
```

6. [11 marks] In class we learned mergesort. Part of the mergesort algorithm merges two arrays that are already sorted into one sorted array. This task is called *two-way merge*, as we merge two sorted arrays into one.

   This question asks you to implement a function that performs 3-way merge. That is, this function merges three sorted arrays (in increasing order), and stores the resulting sorted sequence in a fourth array.

   More precisely, implement a function whose prototype is given below:

```
void merge_3way(int array1[], int len1, int array2[], int len2,
                int array3[], int len3, int array[]);
```

   In this prototype, `array1`, `array2` and `array3` are sorted arrays (in increasing order) of lengths `len1`, `len2` and `len3`, respectively. This function merges them and stores the result in array `array`.

   No error handling is required; you can assume that `array1`, `array2` and `array3` are sorted and `array` is large enough to store the result. Do not provide a `main` function; implement `merge_3way` only. There is an extra page for your solution.

```
void merge_3way(int array1[], int len1, int array2[], int len2,
                int array3[], int len3, int array[]) {
  int i = 0, j = 0, k = 0, l = 0;

  while (i < len1 && j < len2 && k < len3) {
    if (array1[i] <= array2[j] && array1[i] <= array3[k]) {
      array[l] = array1[i];
      i++;
```

```
  }
  else if (array2[j] <= array1[i] && array2[j] <= array3[k]){
    array[l] = array2[j];
    j++;
  }
  else {
    array[l] = array3[k];
    k++;
  }

  l++;
}

while (i < len1 && j < len2) {
  if (array1[i] <= array2[j])
    array[l++] = array1[i++];
  else
    array[l++] = array2[j++];
}

while (j < len2 && k < len3) {
  if (array2[j] <= array3[k])
    array[l++] = array2[j++];
  else
    array[l++] = array3[k++];
}

while (i < len1 && k < len3) {
  if (array1[i] <= array3[k])
    array[l++] = array1[i++];
  else
    array[l++] = array3[k++];
}

while (i < len1)
  array[l++] = array1[i++];

while (j < len2)
  array[l++] = array2[j++];

while (k < len3)
  array[l++] = array3[k++];
```

```
}
```