The University of Calgary

ENCM 339 – Programming Fundamentals
Fall 2016

Instructors: S. Norman, and M. Moussavi

Wednesday, November 2 — 7:00 to 9:00 PM

The First Letter of your Last Name: ☐

Please Print your Name - First Name and Last Name:___________________________________

Please check the box for your lecture section …

Lecture Section L01 (MWF – 1:00 to 1:50 PM) – Dr. Norman: ☐

Lecture Section L02- (MWF – 12:00 to 12:50 PM) - Dr. Moussavi: ☐

Your Signature: _______________________________

Please also write your name and your U of C Student ID at the bottom of the last page

Instructions

- To prevent distracting your fellow students, you may not leave the exam room during the last ten minutes of the exam.
- No electronic calculators or computers may be used during the test.
- This is a closed-book test. You must not refer to books or notes during the test.
- Write legibly! What cannot be read will not be marked.
- If you remove the staple from the paper, write your name at the top of every loose page.
- Attempt all questions.
- Get the easiest marks first! If you find a particular question difficult, return to it after completing easier questions.
- If you write any rough work on the question paper, cross it out with a large "X" and write, "Rough work" beside it. Rough work will not be marked.
- Follow the directions for each problem carefully.

<u>SECTION 1 - Multiple Choice Questions (1 mark each) – Total 5 marks:</u>

Notes: To answer the questions in this section you can make the following assumptions:

- o All necessary header files are included
- o Size of `int` data type is 4 bytes, and size of `double` data type and pointers are 8 bytes.

1. What is the output of the following code fragment?

```
char* s1 = "apple";
char s2[]= "Victor";
int* p = malloc(5 * sizeof(int));
printf("%d %d %d %d", (int) sizeof(s1), (int) sizeof(s2),
                      (int) strlen(s2), (int) sizeof(p));
```

   a. 5 6 5 8
   b. 8 7 6 8
   c. 8 8 4 8
   d. 6 6 5 8
   e. 6 6 6 4
   f. None of the above.

2. What is the output of the following code fragment?

```
int a[] = {44, 88, 66, 22, 10, 12, 5};
int* p1 = a;
int* p2 = &a[6];
printf("%d", (int) (p1 – (p2 + 1));
```

   a. 6
   b. 5
   c. -7
   d. 7
   e. None of the above.

3. What if anything is wrong with the following code fragment?

```
char code[6] = "Apple";
const char* csp = code + 1;
csp++;
*code = csp[0];
*csp = *code;
```

   a. There is a compilation error on the second-last line of the code fragment.
   b. There is a compilation error on the last line of the code fragment.
   c. The fragment will cause the containing program to crash when it runs.
   d. All of the above are correct
   e. None of the above is correct.

4. Consider the following code fragment and select one of the following answers. Assume that the user enters the following text on the keyboard, then presses the Enter key: **`52.75  12.625x`**

```
double d = 0.125;
int i = 42, ns = 99;
printf("please enter two numbers: ");
ns = scanf("%d %lf", &i, &d);
printf("%d %d %f\n", ns, i, d);
```

   a. The program output will be: `0 42 0.125000`
   b. The program output will be: `1 52 0.125000`
   c. The program output will be: `2 52 0.750000`
   d. The program output will be: `2 52 12.625000`
   e. None of the above.

5. What is the output of the following code fragment?

```
char course[] = "ENCM339F2016";
char* sp = course + 2 ;
while(*sp != '1'){
    printf("%c", *sp);
    sp = sp + 2 ;
}
```

   a. ENM339F20
   b. CM33F20
   c. C3921
   d. EC3921
   e. None of the above.

SECTION 2 – Short Answer Questions – Total 12 marks:

1. **(4 marks)** The following partial C program should open a text file called `input.txt,` with several `int` values separated by whitespace, reads one `int` value at a time, and copy each value into a binary file, until the end of the input file is reached. You may assume that the data in the text file are all valid integers and that both files will open successfully without errors. Your job is to fill the blank spaces with correct C expressions.

```
#include <stdio.h>
int main(void) {
  int number, nscan;
  FILE *fp1 = fopen("input.txt", "r");
  FILE *fp2 = fopen("output.dat", "wb");
  while (1) {
    nscan = fscanf(.................... , "%d", ....................);

    if (nscan != ............... ) break;

    fwrite ((void *).................., sizeof(..................), 1, ................);
  }

  fclose(.................);

  fclose(.................);

  return 0;
}
```

2. **(2 marks)** Assume that the macros `FOO`, `BAR` and `QUUX` are not defined in `<stdio.h>`. What is the output of the following program?

```
#include <stdio.h>
#define FOO 47
#define QUUX
int main(void) {
#if defined(QUUX)
    printf("99 QUUX\n");
#ifndef FOO
    printf("FOO\n");
#endif
#if 1
    printf("BAR\n");
#endif
#endif
    return 0;
}
```

Write your answer in the following space:

3. **(4 marks)** Assume that the call to `malloc` succeeds. Draw a memory diagram for `point one` in the space beside the code listing.

4.

```
#include <stdio.h>
#include <stdlib.h>
int glob[3] = { 31, 57 };
int main(void) {
  int loc[3];
  int *dyn =
    malloc(3 * sizeof(int));
  loc[0] = glob[0];
  dyn[1] = glob[1];

  // point one

  return 0;
}
```

5. **(2 marks)** Write a `#define` macro with three arguments, called `IN_RANGE`, that produces a value of 1 if the value of the first argument is greater than or equal to the second argument and less than or equal to the third argument, and otherwise produces a value of 0.
Write your macro in the following space:

SECTION 3 – Functions – 26 marks

Part a (8 marks): Write a function definition based on the given function interface comment.

```
int remove_outliers(int* x, int n, int min, int max);
// REQUIRES: n > 0. Elements x[0] ... x[n-1] exist.
// PROMISES: Elements less than min or greater than max are removed from the array.
//  Let rv be the return value. Then rv == number of elements that are >= min
//  and <= max, and x[0] ... x[rv-1] contain those elements.
// EXAMPLE: n is 5, original values of x[0] to x[4] are 5, -1, 4, 11, 10, min is 0,
//   max is 10. Then rv will be 3, x[0] to x[2] get 5, 4, 10, and it does not matter
//   what is left in x[3] and x[4].
```

Part b (6 marks): In mathematics, a Fibonacci integer sequence may have any two integers for its first two values. After that, each value is the sum of the previous two values. The Fibonacci numbers ...

0, 1, 1, 2, 3, 5, 8, 13, 21, …

are an example of such a sequence. Write a function definition for `is_fibonacci`.

```
int is_fibonacci(const int* x, int n);
// REQUIRES: n >= 3. Elements x[0] ... [n-1] exist.
// PROMISES: Return value is 1 if x[0] ... x[n-1] are the first n numbers of a
// Fibonacci integer sequence.  Otherwise the return value is 0.
```

Part c (6 marks): In the following space write the definition of a function called `count_lower_cases` that receives one argument of type `const char*`, called `str,` which must points to the beginning of a valid C string. The function should return the number of the lower-case characters in the string.
Examples:

Function returns 5 for these example arguments	Function returns 0 for these example arguments
`"AaBbCbBbAbYYYYYYY"`	`"ABCA"`
`"abbbbACCCA"`	`""`
`"ABAbbbbb"`	`"!"`
`"AaabddA"`	`"123445"`
	`"123ABC&%$"`

Important Note: You are **NOT allowed** to use any of the C library functions in your solution. You may assume that the character set is ASCII, in which `'a' == 97, 'b' == 98`, and so on.

Part d (6 marks) Write a function definition to match the following interface.

```
void trim_trailing_spaces(char* str);
// REQUIRES: str points to the beginning of a valid C-string.
// PROMISES: If the string has trailing spaces, the string is shortened to eliminate those
//    spaces.  Otherwise the string is not changed.
// EXAMPLE: Assume each little box, □, represents a space in the following string:
//          char[ ] foo = "AB%□&□98ax□□□□□□□";
//          after the call trim_trailing_spaces(foo), the string should match "AB%□&□98ax"
//          Notice: Only the trailing spaces are removed.
```

SECTION 4 – Arrays and Pointers (10 marks):

Consider the definition of the following program and draw memory diagrams for point one and two in the given boxes to the right of the code:

```
#include <stdio.h>

void saturn(const char *str, char arr[]);

void mercury(int x[], int* y);


int main(void)
{

    char code[] = "APEX";

    const char* exam = "FALL16";

    int array[] = {4, 17, 300, 400};


    saturn(&exam[1] , code + 2);

    mercury(array, array + 2);


    return 0;
}


void saturn(const char *str, char arr[])
{

    arr[0] = *(str-1);

    // point one

}

void mercury(int x[], int* y)
{
    int z = *x;

    *x = y[0];

    y[0] = z;

    // point 2
}
```

Point 1

Point two

SECTION 5 – Structures – Total 10 marks

Consider the following declarations:

```
       typedef struct Point { double x, y; } Point;

       typedef struct Rectangle {
          Point left_top_corner;
          double width, height;
       } Rectangle;
```

Part 1 (4 marks) - In the following space, draw an AR diagram for **point one** in function funct:

```
void funct(Rectangle rect, Rectangle* prect) {
    prect->width = rect.width;
    // point one
}

int main(void) {
    Rectangle a = {{20, 30}, 15, 66}, b = {{10, 20}};
    double y;
    funct(a, &b);
    y = diagonal(&a);
    printf( "diagonal distance between the top-left and bottom-right corners is: %f\n ", y);
    return 0;
}
```

Part 2 (6 marks) – Consider the following function prototype and its interface comment, and write the definition of function **diagonal**. (Hint: Assume that #include <math.h> has been done. `double sqrt(double x);` is a function prototype in that header file.)

```
double diagonal(const Rectangle *rect);
// REQUIRES: rect points to an instance of Rectangle.
// PROMISES: Return value is the diagonal distance between the left-top corner of the
//           rectangle and its right-bottom corner.
```

Your Name and U of C Student ID Number:

Don't write anything in this box – this box is for marking

Section I	Section II	Section III	Section IV	Section V	Total
/5	/12	/26	/10	10	/63