PASS MOCK EXAM – FOR PRACTICE ONLY

Course: **SYSC 2006 C** Facilitator: **Madeline Harlow**
Dates and locations of mock exam take-up: **Monday Feb. 4, 6-730 in SA 520**

IMPORTANT:

It is **most beneficial** to you to write this mock midterm **UNDER EXAM CONDITIONS**. This means:

- Complete the midterm in 1.5 hour(s).
- Work on your own.
- Keep your notes and textbook closed.
- Attempt every question.

After the time limit, go back over your work with a different colour or on a separate piece of paper and try to do the questions you are unsure of. Record your ideas in the margins to remind yourself of what you were thinking when you take it up at PASS.

The purpose of this mock exam is to give you practice answering questions in a timed setting and to help you to gauge which aspects of the course content you know well and which are in need of further development and review. Use this mock exam as a *learning tool* in preparing for the actual exam.

Please note:

- Come to the PASS session with your mock exam complete. There, you can work with other students to review your work.
- Often, there is not enough time to review the entire exam in the PASS session. Decide which questions you most want to review – the facilitator may ask students to vote on which questions they want to discuss.
- Facilitators do not bring copies of the mock exam to the session. Please print out and complete the exam before you attend.
- Facilitators do not produce or distribute an answer key for mock exams. Facilitators help students to work together to compare and assess the answers they have. If you are not able to attend the PASS session, you can work alone or with others in the class.

Question 1 Fundamental Programming Syntax [18 marks]

(a) [4 marks] What is printed out by the following code fragment? (What will you see on the console?) Show your work for partial marks.

(i)
```
int x = 6, y = 10;
do {
y = y – x;
x = x – y;
} while (x > 0);
printf("x = %i y = %i\n", x, y);
```
ANS:_____(1)

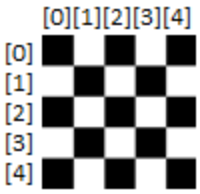
(ii)
```
int a = 3, b = 6;
while (a < 5 && b < 8) {
    a = a + 2;
    b++;
}
printf("a = %i b = %i\n", a, b);
```
ANS:_____(1)

(b) [4 marks] Any quadratic equation can be solved with the quadratic formula. Write a **program fragment** to **prompt-and-input** 3 variables, and then compute and print the two zeros of the quadratic formula. To make things easier, only calculate the roots if they are real or repeated, otherwise output an error (error if roots are complex). State any assumptions you make.

Quadratic Formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

(c) [4 marks] Write the **program fragment** to declare and initialize the checker board shown below.



(d) [6 marks] Write the **function** that receives two integers (a and b) and returns one integer (sum). The function will return the sum of all the integers in between but exclusive of the two integers provided. For example if 3 and 7 are inputted, the function would calculate 4 + 5 + 6 and return the value 15. Sample implementation of this function is provided below. Do not assume that the values passed through the function are in order; in other words, your code must operate when a is greater than b AND when b is greater than a. If a and b are equal, return 0.

```
#include <stdio.h>
int intSum (int a, int b);          //<----------          What is this piece of code called?

int main (void)
{
        int x = 3, y = 8, answer;
        answer = intSum(y, x);
        printf("%d\n", answer);   //<--------          What will be printed to the console?
        return 0;
}

___ intSum (___ a, ___ b)                    //<------------          Start your function here (6)
{
```

Tip: Read the **whole** program first so that you understand the central data structure: a **byte** represented as an array of 8 boolean bits.
Tip: Study the **main()** function to understand how functions fit together.

Question 2: Functions and Arrays [9 marks]

```
#include <stdio.h>
#include <stdbool.h>
#include<math.h>                    //includes the function prototype: int pow(int base, int exponent)
/*
 * @param byte   An array of 8 boolean "bits", each bit element set to either true(1) or false(0)
 *  @return              Return the parity of the byte parameter, where parity is computed from the count of
 *                       true(1) bits within the byte parameter.
 *       If number of true(1) bits is an even number, the parity is even and the function returns true. Otherwise,
 *       the parity is odd and the function returns false. Note: zero is an even number.
 */
bool getParity(bool byte[8])
{       //Insert code using a FOR loop









}
/*      Function runs one test case, comparing the computed parity of the given byte with the expected parity
 *      (that is given as a parameter). An error message is printed only if the test case fails.
 */
void testGetParity(bool byte[8], bool expectedParity)
{       //Insert code here









}
int main(int argc, char* argv[]) {
        bool byte1[8]={false, false, false, false, false, false, false, false};          //Example 1
        testGetParity(byte1, true);
        bool byte2[8]={true, true, true, true, true, true, true, true};                  //Example 2
        testGetParity(byte2, true);
        bool byte3[8]={true, false, false, false, false, false, false, false };          //Example 3
        testGetParity(byte3, false);
        bool byte4[8]={ false, false, false, false, false, false, false, true };         //Example 4
        testGetParity(byte4, false);
        //... and so on..
        printf("Test Complete.\n"); return 0; }
```

Extra Practise: Arrays

1. a.) Write the **program fragment** that declares a character array called “array” with 5 elements all initialized to the character A. (3 marks)

b.) Write the **program fragment** that changes each element of array to the following 5 characters (ignoring commas and blank space): (2 marks)
B, M, 4, !, K,

c.) Write the **program fragment** that uses a loop to set all elements in array to a blank space. (2 marks)

d.) Write the **program fragment** to declare and initialize a 2D integer array using loops (no hard coding) to match the array below.

$$\begin{matrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{matrix}$$

2. a.) Write a program that initializes the integer array, x = [4, 6, 23, -5, 7, 4, 0, 5, 4], and uses a loop to count the number of 4's found in the array. The program should print to the console a statement declaring how many 4's are found in the array. BONUS: Also print to the console the elemental location (index) of each 4 in the array. (5 marks)

b.) Write the **program fragment** that can be added to your above program that will print out the MAXIMUM value found in the array. BONUS: Print out both the Maximum and Minimum values as well as their elemental location (index) in the array. (3 marks)

Extra Practise: Functions

1. Demonstrate how to call the following function: (2 marks)

```
/* Calculates the square of the value provided (x^2)
 * @param     An integer named value
 * @return    Returns the squared value (type int)
 */
int valueSquared (int value)
{
      return value*value;
}
```

2. A function description is shown below.
   (a.) Give the appropriate function prototype
   (b.) Demonstrate how to call the function in main().
   (c.) Write the implementation of the function

```
/* Function: add
 * @param     a     An integer value
 *            b     An integer value
 * @return          Returns the sum of a and b
 */
```

3. Create initialize functions for Question 1C and Extra Practise Array Question 1D. In otherwords, take the code you have written and put it into a function whose purpose is to initialize that array. Then show how to call the function.