

**Programming & Software Development
CSCI 455
SAMPLE MIDTERM**

**Dr. K. Narayanaswamy
Midterm Examination**

You have 75 minutes to complete this examination. OPEN BOOK AND OPEN NOTES:
-- you can use any and all reference materials. (Laptops and cell phones are NOT allowed)

YOU MUST ANSWER **ALL QUESTIONS**. **PART I** (which starts on Page 2 and ends on Page 7) contains very brief questions, mostly of the multiple choice variety, and some that require brief responses within the provided spaces. In **PART II**, which starts on page 8, there are 5 questions in all. These questions require longer responses. If you need more space (which should not normally be the case), make a note that your answer is continued overleaf, and write on the back of the page.

NAME: _____

STUDENT ID: _____

QUESTION	MAXIMUM	SCORE
PART I	50	
PART II	50	
Question 1	10	
Question 2	10	
Question 3	10	
Question 4	10	
Question 5	10	
TOTAL	(100)	

PART I : Requires relatively brief answers. Most are multiple choice. **Unless noted otherwise, each question is worth 1 point.**

Questions 1, 2, 3, 4, 5, 6, 7, 8 and 9 are based on code below:

```
void myfun (int x, int & y)
{
    int temp = x;
    x = y;
    y = temp;
}
```

1. Name a formal parameter in the function above:
2. Name a parameter passed by reference:
3. Name a parameter passed by value:
4. “temp” is best termed:
 - a) Value parameter
 - b) Reference parameter
 - c) Local variable
 - d) none of the above
5. The type of results returned by the function “myfun” is:
 - a) int
 - b) short
 - c) float
 - d) no result returned
6. In the main program, if you have the following code:

```
int a = 3, b=5;
myfun(a, b);
```

the identifiers “a” and “b” are called:
 - a) actual parameters
 - b) local variables
 - c) formal parameters
 - d) local parameters
7. The value of the variable “a” after “myfun” is called in the main program as above:
 - a) 6
 - b) 12
 - c) 8
 - d) 3
8. The value of the variable “b” after “myfun” is called in the main program as above:
 - a) 6
 - b) 12
 - c) 8
 - d) 3
9. Typical C++ programs will have a main() function
 - a. True
 - b. False
10. What kind of loop is best if we wish to execute the loop body at least once before we test the loop condition?
 - a) for
 - b) while
 - c) do/while
 - d) none of these

11. What kind of loop is appropriate most of the time with arrays?
a) for b) while c) do/while d) none of these

12. What is wrong with the following declaration?

- ```
char x = "x";
```
- a) Nothing
  - b) The double quotes should be single quotes
  - c) Characters cannot be initialized
  - d) The word "char" should be replaced by "int"

In the context of the following declaration (assuming <string> and <ctype.h> are included by the programmer) **(for questions 13, 14, 15)**

```
string x = "123Bob", y = "Jane456", z;
```

13. A call "x.empty( )" would yield what value?

- a) true
- b) false
- c) unknown
- d) depends

14. A call "x.size( )" would yield what value?

- a) 4
- b) 5
- c) 6
- d) 7

15. A call "y[4].isdigit( )" would yield what value?

- a) true
- b) false
- c) unknown
- d) depends on circumstances

16. Write a function as below: **(4 points)**

```
// returns true if ALL the characters in s are uppercase alphabetic characters;
// false otherwise
bool all_uppercase (string s)
{
```

```
}
```

17. What is the output of the following code segment: **(2 points)**

```
for (int i =1; i <10; i++)
{
 cout << i;
 if (i >=5) break;
 cout << i;
}
```

- a) 112233445566778899
- b) 12345
- c) 11223344
- d) 112233445

Consider the following code: **(for questions 18, 19, 20, and 21)**

```
const int size = 7;
int scores[size][size];
```

18. The number of elements in the scores array is:

- a) 64
- b) 49
- c) 63
- d) 48

19. The square array above has the same number of rows and columns. A square has 2 diagonals, connecting opposite corners of the square. Use **nested for loops** to make all the elements (i.e., array cells) that are along both diagonals equal to 0. An example of a square table and the diagonal is shown in the diagram above: **(5 points)**

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 0 |   |   |   |   |   | 0 |
|   | 0 |   |   |   | 0 |   |
|   |   | 0 |   | 0 |   |   |
|   |   |   | 0 |   |   |   |
|   |   | 0 |   | 0 |   |   |
|   | 0 |   |   |   | 0 |   |
| 0 |   |   |   |   |   | 0 |

20. Let us say that each row of the scores array represent a player on a team. Write only the code to compute the average of each player's scores, **(3 points)** and print correctly on separate lines **(1 point)** the average score for each of the players on the team.

21. The function `read_scores` is supposed to take as parameters the file name from which to read the scores for each player and an array in which it can return the scores read from the file. It should return a result indicating whether the file was successfully read or not. Assume that the file has information formatted appropriately and has the right number of items, define the entire `read_scores` function below: **(4 points)**

```
bool read_scores (int scores[size][size], string inputfilename)
{
```

```
}
```

Questions **22 and 23** use the following:

```
enum people { PROFESSOR, STUDENT, STAFF };
```

22. The above declares an enumerated type. The benefit of this type is that:
- a) Program is easier to understand
  - b) Program executes faster
  - c) Program is longer
  - d) Program is shorter
23. The code: `cout << STUDENT;` would produce as output:
- a) STUDENT
  - b) 0
  - c) 2
  - d) 1
24. Supposing we had to model all the properties of a Student at a university, such as social security number (which is a string), name (also a string), age (an integer), and address (which is a string). Write down a type declaration in C++ appropriate to model Student objects: **(2 points)**

25. Complete the function below that returns “true” if the incoming parameter is a valid social security number (something that conforms to the format XXX-XX-XXXX where each of the X’s is a digit, with hyphen or dash in the place shown), and false otherwise.

**(5 points)**

```
bool is_valid_ssn (string ssn)
{
```

```
}
```

26. The function `initialize_student` is intended for users to provide attributes to initialize any student object that is provided as the first parameter. It will pass back “true” if the object is properly initialized, and “false” if any of the attributes are invalid. Given that description, is there an error in the prototype of `initialize_student` below? (1 point). Explain your answer adequately (1 point).

```
bool initialize_student (string ssn, string name, string address, int age, Student x);
```

27. Write the code for the function “`initialize_student`”, with any prototype corrections if required. You must ensure that the social security number is valid and that the age is between 0 and 100 (inclusive of both). You should use the function “`is_valid_ssn`” from Question 25. If all the values provided are valid, initialize the object `x` and pass back true. If **any** of the incoming values are invalid, **no changes** should be made of any kind to the object, and the result passed back should be “false”. (4 points)

**PART II:** Requires hand simulation skills and possibly the writing of small code segments. Your answers should fit into the space provided. Otherwise, indicate that you are continuing on the backside of the page, and finish your answer there. There are 5 questions in this part. Each carries 10 points.

**PART II: Question 1 (10 points)** You are given the following functions:

```
// returns true if "n" is a fibonacci number; false otherwise
bool fibonacci (int n) { ... }
```

a) Fill in the body of the function `is_prime` below: **(4 points)**

```
bool is_prime (int n) // returns true if n is a prime number, false otherwise
{
```

```
}
```

b) Fill in the body of the following function: **(3 points)**

```
bool both_prime_and_fibonacci (int n)
{
```

```
}
```

c) Fill in the body of the following function: **(3 points)**

```
bool prime_but_not_fibonacci (int n)
{
```

```
}
```

**PART II: Question 2 (10 points)** You are given the following declaration:

```
int A [7];
```

- a) **After executing the following code**, show the state of the array using as many cells as you think you need: **(4 points)**

```
for (int l = 6; l >= 3; l--) A[l] = 6 - l;
for (int l = 0; l < 3; l++) A[l] = l;
```

Array A:

|         |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| INDEX → |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| VALUE → |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

- b) Write down the output of the following code assuming the array is in the above state:

```
int result = 0;
for (int n = 6; n > 0; n--)
{
 result = A[n] + A[n-1] + result;
 cout << "Running result: " << result << endl; // Line 1
}
```

Show the values of the variables below upon execution of each iteration of the loop **using as many cells as you need** below: **(6 points)**

|                  |  |
|------------------|--|
| n = →            |  |
| result = →       |  |
| TerminalOutput → |  |

|                  |  |
|------------------|--|
| n = →            |  |
| result = →       |  |
| TerminalOutput → |  |

|                  |  |
|------------------|--|
| n = →            |  |
| result = →       |  |
| TerminalOutput → |  |

|                  |  |
|------------------|--|
| n = →            |  |
| result = →       |  |
| TerminalOutput → |  |

|                  |  |
|------------------|--|
| n = →            |  |
| result = →       |  |
| TerminalOutput → |  |

|                  |  |
|------------------|--|
| n = →            |  |
| result = →       |  |
| TerminalOutput → |  |

|                  |  |
|------------------|--|
| n = →            |  |
| result = →       |  |
| TerminalOutput → |  |

**PART II: Question 3 (10 points)** Given the array declaration as below:

```
const int size = 10;
int A [size];
```

- a) Why is it better to define the array size using a constant? **(1 point)**
- b) Write the body of the following function. “m” is the smaller index and “n” is the higher index. You should return a value of ERROR (assumed to be a global integer constant) if m or n is invalid in any way or if m is larger than n. **(6 points)**.  
// returns the minimum of the numbers of the array “input” from index m to n (including n)  
int minimum (int input[size], int m, int n)  
{

}

- c) What is the terminal **output** produced by the following code: **(3 points)**

```
for (int n= 0; n < size; n++) A[n] = size - n ;
cout << “Minimum: “ << minimum(A, 4, 7) << endl;
```

Use the working area below **IF YOU WISH** to show the contents of Array A:

|         |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| INDEX → |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| VALUE → |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**PART II: Question 4 (10 points)**

Consider the code segment below:

```
string temp;
ifstream inputfile;
ofstream outputfile;
inputfile.open("input.txt");
outputfile.open("output.txt");
while ((inputfile >> temp) != NULL) { outputfile << temp; outputfile << endl; }
```

AND, given the contents of the file "input.txt" as below:

There are 99 different emergency  
broadcast systems in the 90043 zip code!!!

- a) Write down the **exact** contents of the file "output.txt". AFTER executing the code above (4 points)

- b) Write the body of the function called *file\_length* below that accepts a filename as parameter, opens the file, and returns as result the length of the file in characters (i.e., every character read from the file, regardless of what the character is, increments the length by 1). If the file does not exist return -1. (6 points)

```
int file_length (string inputfilename)
{
```

```
}
```

**PART II: Question 5 (10 points)**

```
#include <iostream>
using namespace std;

void swap (int &a, int &b)
{
 int temp = a ;
 a = b ;
 b = temp ;
}

void mystery (int elements[], int low, int high)
{
 if (low >= high) return;
 swap(elements[high], elements[low]);
 mystery(elements, low + 1, high - 1);
}

main ()
{
 int collection[5];
 for (int index = 0; index < 5; index++) collection[index] = index ;
 for (int index = 0; index < 5; index++) cout << collection[index] << " " ; // Line 1
 cout << endl;
 mystery(collection, 1, 3) ;
 for (int index = 0; index < 5; index++) cout << collection[index] << " " ; // Line 2
}
```

- a) What is the output of Line 1 of the program above? **(4 points)**  
Use the working area below IF YOU WISH to note the contents of the array:

|         |  |  |  |  |  |  |  |  |
|---------|--|--|--|--|--|--|--|--|
| INDEX → |  |  |  |  |  |  |  |  |
| VALUE → |  |  |  |  |  |  |  |  |

- b) What is the output of Line 2 of the program above? **(6 points)**  
Use the working area below IF YOU WISH to note the contents of the array:

|         |  |  |  |  |  |  |  |  |
|---------|--|--|--|--|--|--|--|--|
| INDEX → |  |  |  |  |  |  |  |  |
| VALUE → |  |  |  |  |  |  |  |  |