**CptS 121 – Program Design and Development**

**Spring 2021**

***Take-home Exam 1***

Friday, February 19, 2021

**Your Name: \_\_\_\_\_\_\_Duy Pham\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ID Number: \_\_\_\_\_011742984\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**TA’s Name: \_\_\_\_\_Aayush Bajoria\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

READ THE FOLLOWING INSTRUCTIONS:

This exam is take-home. It should take ~ 1 hour to complete. You must work **individually** on this exam. You may use your book, notes, and online resources if necessary. Please either handwrite or type your answers into this document. Late exam solutions will **not** be accepted! **Please show all work!!!**

You must submit the exam through Blackboard by Monday, February 22, midnight PST. Be sure to answer each question precisely. Do not provide superfluous details in your answers. *NOTE: you do not need to comment your code solutions.*

|  |  |  |
| --- | --- | --- |
| **Part** | **Points Possible** | **Points Earned** |
| I | 50 |  |
| II | 50 |  |
| **Total** | **100** |  |

***Part I: Conceptual Questions (50 pts) – Short answer, Fill-in-the-blank, and True/False***

1. (3 pts) What is a *function*? Explain.

A function is a block of code that takes inputs, does some specific computation and

produces output. Each functions helps solve a subproblem discovered when we

break a large, complex solution into logical units. We also call the function to avoid

writing the same code again and again for different inputs.

1. (3 pts) Why do we *comment* our programs? Explain.

Comments are necessary to explain our code. We are always supposed to program

and work together with others in a team. So comments can make the source code

easier for others to understand. Besides, if we review the code later, comments

may help us memorize our logic that we have written before.

1. (3 pts) What is a *structure* *chart*? Explain.

A structure chart is a chart showing the breakdown of a problem to small parts. It is

a top-down modular design tool into a tree. This is a good way to analyze the

connection between subactivities for an easier understanding.

1. (3 pts) What is *ASCII*? Please elaborate.

ASCII stands for American Standard Code for Information Interchange. Computers

can only understand numbers, so an ASCII code numerically represents a character

such as 'a' or '@' or an action of some sort. ASCII encodes 128 specified characters

into seven-bit integers. Ninety-five of the encoded characters are printable: these

include the digits 0 to 9, lowercase letters a to z, uppercase letters A to Z, and punctuation symbols.

1. (4 pts) Is the following an *algorithm*? Why or why not?

* Cut two pieces of bread
* Place the bread in a toaster
* Take the bread out of the toaster
* Spread peanut butter and jelly on the bread
* Repeat

This process is not an algorithm. Because it does not stop after a finite amount of

time despite of being well ordered.

1. (6 pts) Given a blue whiteboard marker and a whiteboard, provide an algorithm to describe to another human how to draw a *rectangle* on the whiteboard. Be sure to number your steps. You may or may not need all of the lines provided below.

+ List of steps:

- Step 1: Take the blue whiteboard marker in hands.

- Step 2: Draw a line segment of arbitrary length vertically from top to bottom.

- Step 3: Right at the end of the first line segment, turn right 90 degrees and draw

the second line segment with an arbitrary length.

- Step 4: Right at the end of the second line segment, turn above 90 degrees and

draw the third segment of the same length as the first one.

- Step 5: Right at the end of the third segment, turn left 90 degrees and connect

this point to the start point of the first segment.

+ The process finishes and stops.

1. (2 pts) List two advantages of the 3-file format/organization discussed in class.
2. We can manage functions aside the main function, which is easier to look out and track down everything in the program.
3. We place our code into more manageable pieces, which may create our own libraries that could import many different projects.
4. (2 pts) Given the following function. Are the explicit typecasts required in the computation? Explain.

double calculate\_parallel\_resistance (int r1, int r2, int r3)

{

double parallel\_resistance = 0.0;

parallel\_resistance = 1 / ((double) 1 / r1 + (double) 1 / r2 + (double) 1 / r3);

return parallel\_resistance;

}

Yes, the explicit typecasts are required. Because r1, r2, r3 and integer 1 are integer

values, 1/r1, 1/r2 and 1/r3 will yield int values instead of double. So we need the

integer 1 to explicitly cast to type double to get the more precise result (a floating-point value).

1. (2 pts) Function \_\_\_ fprintf ( )\_\_\_\_ is used to write data to a file.
2. (2 pts) A(n) \_\_\_\_\_formal\_\_\_\_\_\_\_\_\_\_\_\_\_ parameter is used in a function definition to represent the position and type of a corresponding actual argument.
3. (2 pts) What C data types should be used to represent the following? You may list *int*, *double*, or *char* only.

The average number of people in a household in the world? \_\_\_\_\_double\_\_\_\_\_\_

1. (2 pts) Evaluate the following C expression (if no decimal point is listed, the value is an integer):

3 % 2 + 6 / 7 – 10.5 \* 1 = \_\_\_\_\_\_-9.500000\_\_\_\_\_\_\_\_\_\_\_\_

1. (6 pts) Rewrite the if statement below using only the relational operator < in all conditions. Assume that the value of score is between 0 and 100 inclusive.

|  |  |
| --- | --- |
| Original if statement:  if (score >= 90)  printf("A\n");  else if (score >= 80)  printf("B\n");  else  printf("C\n"); | Your rewrite using only <:  if (score < 80)  printf(“C\n”);  else if (score < 90)  printf(“B\n”);  else  printf(“A\n”); |

1. (2 pts) True / False: When we implement code, we strive for *highly cohesive* functions. (True)
2. (2 pts) True / False: An identifier/variable name, in C, may NOT start with a digit.

(True)

1. (2 pts) True / False: The numerical code that represents ‘B’ is greater than the code that represents ‘a’. (False)
2. (2 pts) True / False: Logic errors are reported by the compiler. (False)
3. (2 pts) True / False: A function *prototype* is a declaration to compiler of a newly defined function; name, order of arguments and corresponding types, and return type are very important. (True)

***Part II: Programming Questions (50 pts) – C code***

1. (10 pts – 1 pt/error) Circle or underline all *ten* syntax, logic, and runtime errors found in the following C fragment. Be sure to circle omitted keywords, values, and punctuation. Be sure to correct the errors.

// Prototypes

// The function volume\_cone () computes the volume of cone defined

// by the arguments.

double volume\_cone (double radius, double height);

int main void);

{

double volume = 0.0, r = 0.0, h == 0.0;

print ("Enter the radius and height (both floating-point) of a cone: ");

scanf ("%lf%d", &r );

volume\_cone = volume\_cone (r, height);

printf ("The volume is: $.2lf\n", volume);

return 0

}

… // The volume\_cone() definition would go below here.

+ Correct the errors:

1. Fill (
2. Delete ;
3. Delete =
4. Fix into printf
5. Change d into lf
6. Fill , &h
7. Fix into volume
8. Change height into h
9. Change $ into %
10. Fill ;
11. (6 pts) Write a function called read\_character() that accepts a handle to an already opened file (FILE \* type) as a parameter. This function reads one *character* from the file and returns it.

char read\_character (FILE \* type)

{

char character = NULL;

fscanf (type, “ %c”, &character);

return character;

}

1. (10 pts) Write a function speed\_of\_sound() that calculates the speed of sound (a) in air of a given temperature T (degrees F). The function should accept T as an *integer* parameter. Use the formula:

a = 1086 ft \* (sqrt ((5T + 297) / 247))

Be sure your program does not lose the fractional part of the quotient in the formula shown. Return the calculated speed as a *double*.

double speed\_of\_sound (int T)

{

double a = 0.0;

a = 1086 \* (sqrt ( (double) (5\*T + 297) / 247 ));

return a;

}

1. (12 pts) We need to hire software engineers for our company. We have created an elaborate method for assigning a score to each candidate. Essentially the method takes into consideration the experience, number of previous projects completed, and recommendations received for each candidate. The score assigned to each candidate is 1 – 5, inclusive. Luckily, a function has already been developed to handle the scoring. We just need to write a function called best\_fit (), which does not return a value, but accepts the floating-point scores for three different candidates as arguments. The function must compare the scores and determine, which candidate is the “best” fit for our company. The highest score is considered the “best” fit. The function must print out a statement such as “Candidate x, with a score of y is the best fit!”, where x is 1, 2 or 3 and y is 1 - 5. Be sure to write the function header!

Precondition: the three scores must be unique and between 1 – 5, inclusive.

void best\_fit (double y1, double y2, double y3)

{

double max = y1;

if (max < y2) {max = y2;}

if (max < y3) {max = y3;}

if (max == y1)

{ printf(“Candidate 1, with a score of %lf is the best fit!\n”, y1); }

else if (max == y2)

{ printf(“Candidate 2, with a score of %lf is the best fit!\n”, y2); }

else { printf(“Candidate 3, with a score of %lf is the best fit!\n”, y3); }

}

1. (12 pts) Write a function called is\_vowel() that accepts a *character* asan input parameter, and returns 1 (true) if the character is a vowel (a, e, i, o, u) and 0 (false) otherwise. Hint: you must check for both uppercase and lowercase characters. You may NOT use any functions from <ctype.h>.

int is\_vowel (char character)

{

switch (character)

{

case 'a':

case 'A':

printf("The character is a vowel!"); return 1; break;

case 'e':

case 'E':

printf("The character is a vowel!"); return 1; break;

case 'i':

case 'I':

printf("The character is a vowel!"); return 1; break;

case 'o':

case 'O':

printf("The character is a vowel!"); return 1; break;

case 'u':

case 'U':

printf("The character is a vowel!"); return 1; break;

default: printf("The character is not a vowel!"); return 0; break;

}

}