Anwar Adam

Alan Palayil

Rezwan Rifat

**CS 115 Fall 2019 Lab #2**

Due: **Thursday, September 12th, MIDNIGHT**

Points: **20**

**Instructions:**

1. Use this document template to report your answers. Enter all lab partner names at the top of the first page.

2. You don’t need to finish your lab work during the corresponding lab session.

3. Name the complete document as follows:

LastName\_FirstName\_CS115\_Lab2\_Report.doc

4. Submit the final document to Blackboard Assignments section before the due date. No late submissions will be accepted.Lab Partner 1 Name

Lab Partner 2 Name

**Objectives:**

1. (10 points) Demonstrate the ability to break a basic problem down into inputs, process and outputs.

2. (10 points) Demonstrate the ability to design test cases for your problem.

**Problem 1:**

Break a basic problem down into **inputs**, **process** and **outputs** and write **pseudocode** (step by step sequence of necessary actions) to solve the problem. **NO Java CODE IS NEEDED**

Answer the following questions for the problems listed below. Populate provided tables (enter as many rows as you find necessary) and pseudocode boxes with your answers. Feel free to add extra tables, boxes, comments, etc. if needed.

INPUTS: What are the inputs?

n  What format / data type are they? (integer, real number, single character, string - a sequence of characters)

n  Any valid / invalid / illegal / special values? (positive, negative, valid range, etc.)

n  How do you get them? (enter manually, ask user, read from file, etc.)

PROCESS: How do you get from inputs to the outputs you want?

n  What are the calculation steps?

n  To follow these steps, what else do you need? (formulas, etc.)

n  Other variables, constants, conversions (besides input and output variables)

OUTPUTS: What are the outputs?

n  What format / data type are they in? (integer, floating-point, character, or string)

n  Any valid / invalid / illegal / special values? (positive, negative, valid range, etc.)

n  How do you output them? (display on screen, save to a file, plot, tabularize, etc.)

1. A teenager gets retained by a neighborhood association to distribute fliers, collect dues, and do miscellaneous chores. Just to make sure he is around when needed, he gets 40 dollars a month (he doesn't have to work for that). Plus he gets $11.25 per hour for any time he actually works in a month. Calculate how much the teenager earns if he works H hours in a month. **[2.5 points]**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Inputs and outputs (use “N/A”, “undefined”, “none”, etc. If necessary)** | | | | | |
| Variable name | Input or  Output? | Data type / format | Constraints | Special cases | Comments |
| Teenager (t) | input | char | Only one person is being calculated | N/a | We are calculating the potential income of a teenager |
| Monthly (m) | input | The data type is integer | The amount is $40 and it is given once a month | The amount cannot be different than $40 | This is the guaranteed income for the teenager, regardless of the time spent working. |
| Hours (h) | Input | The data type is integer | The amount given is hours spent working | The value can range from 0 to 720 | This is the amount of time the teenager spends working |
| Potential income (p) | output | The data type is integer | The amount is dependent on the amount of time spent working. | The range of the potential income is $40 to $8140 | This is the potential income for the teenager, and it is figured out by multiplying the hours spent working times the hourly wage, plus the monthly payment. |
| Wage (w) | input | The data type is integer | The amount is constant and it is $11.25 per hour | N/a | This is how much the teenager will earn per one hour of work. |
|  |  |  |  |  |  |

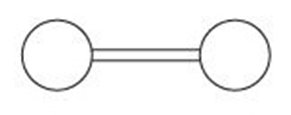
|  |
| --- |
| **Pseudocode:** |
| To find the income of a teenager for doing chores is calculated by using the code below.   1. Teenager will earn $40 per month, regardless of the amount of time spent working. 2. The teenager will earn $11.25 per hour (actual work), and multiply 11.25 \* the hours spent working. 3. If the teenager worked 20 hours a month, multiply 11.25(20) = $225. 4. Add $40 to the $225 to calculate the total income for the teenager for the month. 5. The teenager earned $265 for working 20 hours during the month.      P = h(w) + m   P = 20(11.25) + 40   P = 265 |

2. Consider the 3 dimensional barbell shown below.

Now:

a) Find the volume of the figure if the radius of each sphere is given, and the length of the bar connecting them is given, and the diameter of the bar is given (**all in the same units**),

b) Find the surface area of the figure. **[2.5 points]**

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Inputs and outputs (use “N/A”, “undefined”, “none”, etc. If necessary)** | | | | | |
| Variable name | Input or  Output? | Data type / format | Constraints | Special cases | Comments |
| Cylinder (c) | input | The data type is float | The value can’t be negative | The value has to be greater than 0 | This will give us the surface area of the cylinder in the barbell, using the formula 2pi rh+2pi r^2 |
| Sphere (s) | input | The data type is float | The value can’t be negative | The value has to be greater than 0 | This will give us the surface area of the sphere, using the formula 4 pi r^2 |
| Total surface area (ts) | output | The data type is float | The value can’t be negative | The value has to be greater than 0 | To find the total surface area, add both the cylinder and the sphere values. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |
| --- |
| **Pseudocode:** |
| Radius of the sphere is 5in, length of the cylinder is 10in, and the diameter is 4in.   1. Find the surface area of the cylinder using the formula sa= 2pi rh+2pi r^2 2. Plug in the values for the cylinder and find the outcome. c = 150.8in. 3. Find the surface area of the sphere using the formula sa = 4 pi r^2 4. Plug in the values for the sphere and find the outcome. s = 314.16in. 5. Find the total surface area of the barbells using the formula, ts = 2(s) + c, the total surface area is 779.12in^2 |

3. Most stop-watches allow you to display the time elapsed as a number of seconds, or as hours:minutes: seconds. So 5437 seconds or 1 hour:30 minutes:37 seconds Given an integer number of seconds, calculate the equivalent integer number of hours, integer number of minutes, and integer number of seconds. **[2.5 points]**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Inputs and outputs (use “N/A”, “undefined”, “none”, etc. If necessary)** | | | | | |
| Variable name | Input or  Output? | Data type / format | Constraints | Special cases | Comments |
| Seconds [s1] | Input | Integer | Seconds can’t be less than 0, [s1>=0] | N/A | If the time is negative, then the value input is invalid. |
| Hours [h] | Output | Integer | Hours can’t be less than 0, [h>=0] | N/A | Hours are calculated by the division of the number of Seconds/3600 |
| Variable [r] | Output | Integer | Variable [r] can’t  be less than 0, [r>=0] | N/A | Variable is used as an intermediate to calculate the remaining seconds from hours to minutes.  r=s%3600 |
| Minutes [m] | Output | Integer | Minutes can’t be less than 0, [m>=0] | N/A | Minutes are calculated by the division of variable by 60.  m=r/60 |
| Seconds2 [s2] | Output | Integer | Seconds can’t be less than 0, [s2>=0] | N/A | Since final output also requires seconds we use another variable s2 to find the remaining seconds from minutes.  s2=r%60 |
| Time [t] | Output | Integer | Time can’t be less than 0, [t>=0] | N/A | Time [t] is the variable that holds the entire time spilt in hours, minutes, and seconds. |

|  |
| --- |
| **Pseudocode:** |
| 1. Input the value of seconds. 2. To find the amount of hours, we use the formula h=s1/3600. 3. With that seconds is converted to hours. 4. Then use a variable r to assign the value of m. 5. r=s1%3600 6. So m=r/60 7. With that the remaining seconds to converted to minutes. 8. To find the remainder seconds, s2=r%60. 9. The total time t= h hrs “ ” m mins “ “ s2 secs. |

4. A person suffering from type II diabetes injects insulin based on measurements of their blood sugar level. If the blood sugar level is less than 115, they don't need to inject any insulin at all. For a value of 115, they inject 1 unit of insulin. For every additional increase of 20 in her blood sugar level, they get one additional unit of insulin. (Thus, for a blood-sugar level of 134, they get 1 unit; for a blood-sugar level of 135, they get 2 units.) Create a formula for calculating the insulin injections and output a table that shows the number of units of insulin injected for a user input range of blood sugar values from 115 upwards. **[2.5 points]**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Inputs and outputs (use “N/A”, “undefined”, “none”, etc. If necessary)** | | | | | |
| Variable name | Input or  Output? | Data type / format | Constraints | Special cases | Comments |
| Sugar level [s1] | Input | Integer | Sugar level can’t be less than or equal to 0 [s1>0] | N/A | Sugar level [s] is the input data and is the determining factor for insulin injection, |
| Insulin [i] | Output | Integer | Insulin [i] is 0 till s<115 units. | For s=115, i=1; and for an increase of 20 units of s from 115, i++ | Insulin injection [i] and value starts from 0. |
| Standard level [s0] | Input | Integer | Standard level [s0] is equal to 115 units. | N/A | It is used to check the difference of sugar level. |
| Variable [s] | Output | Integer | Variable [s= s1-s0] | N/A | It is used to get the difference between the sugar level and standard level, to help calculate the number of insulin injections. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |
| --- |
| **Pseudocode:** |
| 1. Take the input of the sugar level [s1] and check it with standard level [s0]. 2. Variable s= s1-s0, is used to find the difference between the sugar level and standard level 3. For the number of insulin injections i={s}%20 4. So insulin injection is determined using the above formula. |

**Problem 2 (10 points):**

Develop a test plan (a set of test cases) for the following problem. The goal of testing is to determine if the solution (a computer program) to your problem:

n  Behaves correctly / produces correct results when given legal input values,

n  Handles illegal input values correctly (for example: preventing division by zero),

n  Behaves as planned when inputs are assigned “boundary” values.

In simple words, your test plan, at the very least, should:

n  Consider “what could go wrong”,

n  Check if your problem solution behaves as expected.

Also, don’t forget to make sure that you are using correct data types, for example: **number of children HAS TO be an INTEGER and you should have a test case that verifies that!**

**Be VERY thorough!**

**NO Java CODE IS NEEDED**. Figure out the calculation process first (feel free to include it as pseudocode. It is not necessary but may help to understand your thought process) and come up with test cases.

1. The bank wants to be sure you can afford to pay them back before they give you a mortgage. One way they consider your ability to repay is by making sure your total debt doesn't exceed a certain percentage of your income, usually 36-42%. This percentage is called the debt ratio. Given your monthly income (a real number) and other monthly debt payment(a real number), you can use the following formulas to determine the lower and upper limits on your monthly mortgage payments **[2 points]**:

n  Lower limit = (36% of your income) minus your other monthly debt

n  Upper limit = (42% of your income) minus your other monthly debt

|  |  |  |
| --- | --- | --- |
| **Test case name (ex. “negative height”, “typical conditions”, etc.)** | **Input data set for this test case** | **Explain why you chose this test case** |
| Control Test | Income is a positive number. Other Monthly debt is less than income but greater than 0. Lower Limit= 36% | I chose this test case, as this suits the situation of an average person. |
| High debt situation | Income is a positive number. The value of other monthly debt is greater than 50% of income, upper limit = 42% | I chose this situation, as this suits the situation of a high debt situation. |
| Low debt situation | Income is a positive number. The value other monthly debt less than 50% of income, upper limit = 36% | I chose this situation, as this suits the situation of a low debt situation. |
| Invalid debt situation | Either income, monthly debt, upper/ lower limit be less than or equal to 0 | I chose this situation, as this suits the situation of a invalid debt situation. |
|  |  |  |
|  |  |  |

2. A jet aircraft is flying 'height' feet above a level plain at 'speed' mph. Suddenly the ground begins to rise at a 4 degree slope. Calculate the amount of time in seconds (to two decimal places) the pilot has to raise the nose before the aircraft strikes the ground. **[2 points]**:

|  |  |  |
| --- | --- | --- |
| **Test case name (ex. “negative height”, “typical conditions”, etc.)** | **Input data set for this test case** | **Explain why you chose this test case** |
| Control Test | Height, speed be positive numbers greater than 0. Angle of Elevation is slowly increasing to 4 Degree. | I chose this test case, as this suits the situation of an average time taken for nose raised by the pilot. |
| High raise 1 | Speed is constant. Height and angle of elevation increase rapidly. | I chose this test case, as this suits the situation of having the nose raised rapidly by the pilot in a short time. |
| Low raise 1 | Speed is constant. Height and angle of elevation is increasing slowly. | I chose this test case, as this suits the situation of having the nose raised slowly by the pilot in a  long time. |
| High raise 2 | Height, speed and angle of elevation increase rapidly. | I chose this test case, as this suits the situation of having the nose raised rapidly by the pilot in a short time. |
| Low raise 2 | Height, speed and angle of elevation is increasing slowly. | I chose this test case, as this suits the situation of having the nose raised slowly by the pilot in a  long time. |
| Invalid raise | Height, speed and angle of elevation has a negative value. | I chose this test case, as this suits the situation of having the nose raised by pilot would be invalid. |

3. You are planning a picnic. Given the number of children and adults attending the picnic, compute the number of full 1.0 lb packages of hamburger needed (you can only purchase full 1.0 lb packages) **[2 points]**. You know:

n  Children eat one burger,

n  That adults eat two burgers,

n  That each burger weighs ¼ pound,

n  Hamburger comes only in one pound packages.

|  |  |  |
| --- | --- | --- |
| **Test case name (ex. “negative height”, “typical conditions”, etc.)** | **Input data set for this test case** | **Explain why you chose this test case** |
| Control Test | Child= 2,Adult=1, Weight of each burger=0.25 pounds, Weight of one package =1 pound | The most basic of input thus to compare the result to the other case |
| Odd number of children | Child= 3,Adult=1, Weight of each burger=0.25 pounds, Weight of one package =1 pound | To see the error if detected as the answer will not come in complete 1 lb packages of burgers. |
| Negative number of children and/or adult | Child= -2,Adult=-1, the Weight of each burger=0.25 pounds, Weight of one package =1 pound | To see if error is detected as answer will be correct but input is wrong as number of children or adults can  not be negative. |
| Negative weight of burger | Child= 2,Adult=1, Weight of each burger=-0.25 pounds, Weight of one package =1 pound | To see if error is detected as weight can not be negative. |
|  |  |  |
|  |  |  |

4. Given the constant flow rate of the faucet into the sink in volume/sec, the volume of the sink, and the constant drain rate of the drain in volume/sec, determine if the faucet is left running when (if ever) the sink will overflow. Output when it will overflow in seconds, or a message stating it will not overflow. **[2 points]**:

|  |  |  |
| --- | --- | --- |
| **Test case name (ex. “negative height”, “typical conditions”, etc.)** | **Input data set for this test case** | **Explain why you chose this test case** |
| Control Test | Volume=20, Flow rate in volume/sec=10, Drain rate in volume/sec=10 | To compare this to other sets of data as this is a correct set of inputs. With no overflow |
| Drain>Flow Test | Volume=20, Flow rate in volume/sec=5, Drain rate in volume/sec=10 | To confirm that it will never overflow. |
| Flow>Drain Test | Volume=20, Flow rate in volume/sec=10, Drain rate in volume/sec=5 | To confirm that the message of overflow will be sent. |
| Negative Values | Volume=-20, Flow rate in volume/sec=10, Drain rate in volume/sec=10 | To see if error is shown as volume cannot be negative |
|  |  |  |
|  |  |  |

5. One of the more interesting statistics for comparing power hitters in baseball is slugging percentage. A hitter's slugging percentage is calculated as follows **[2 points]**:

slugging percentage = (singles + 2\*doubles + 3\*triples + 4\*homeruns) divided by at bats

|  |  |  |
| --- | --- | --- |
| **Test case name (ex. “negative height”, “typical conditions”, etc.)** | **Input data set for this test case** | **Explain why you chose this test case** |
| Control Test | singles,doubles, triples,and homeruns are all positive numbers greater than or equal to 0 and bats is a positive number and  not equal to 0. | I chose this test case, as this suits the situation of an average baseball hitter. |
| High slugging  percentage | singles,doubles, triples,and homeruns are all positive numbers greater than 0 and bats is a positive number and not equal to 0. | I chose this situation, as this suits the situation of a high slugging percentage. |
| Low slugging percentage | Either singles,doubles, triples,and homeruns are all positive numbers but equal to 0 and bats is a positive number and not equal to 0. | I chose this situation, as this suits the situation of a low slugging percentage. |
| Invalid slugging percentage | singles,doubles, triples,and homeruns are numbers less than or equal to 0 and bats is a negative number or  equal to 0. | I chose this situation, as this suits the situation of a invalid slugging percentage which states there is an input error. |
|  |  |  |
|  |  |  |