COMP 10001 Programming Fundamentals
Final Project – Winter 2020

This assignment is worth 20% of your final grade in Programming Fundamentals. It is expected that it will require significant effort on your behalf to complete.

Any form of academic dishonesty will not be tolerated and will be pursued with penalties up to and including a 0 **for the entire course**.

See eLearn for due dates. No late submissions will be accepted.

Introduction

Real world software rarely fits into a single module. Interconnected parts have to work together to solve problems and respond dynamically to user activity. You will create a small game for 1 player that involves simulating rolling dice. The rules are described below and you have been provided with some starter code to help structure your solution.

Everything you need to know in order to complete this assignment is contained within the course material. If you are unsure how to proceed, please engage with your professor, ask questions, and learn from the problem solving efforts.

It is highly suggested that you do not attempt to write this program from start to finish without planning ahead. Taking each module as its own assignment, completing that, and then moving to the next is more likely to give you a satisfactory result.

The assignment will be submitted in 2 parts: your planning and design followed by your Python code implementation. Part of your evaluation will consist of how well these two parts resemble each other.

In order to practice the skills you have been learning and so that you can experience the importance of design in software development, you will be working on a project involving multiple parts.

In creating your program take care to validate all inputs for type and range correctness. Any invalid inputs should produce an error message and you must repeat the input until a valid value is provided. Inputs should not be case sensitive. For example, if a prompt requires a yes or no answer, you should accept Yes, yEs, yeS, YEs, YeS, yES, yes, YES, no, No, nO, and NO. Ideally you would make use of string manipulation to accomplish this.

The sample output is simply that, an example. It is provided to help you understand how the program works. You do not need to produce the exact same output but you must correctly represent the logic described.

The Game

The game will consist of a number of turns that varies from game to game. The user can continue to play the game until they choose to stop. Keep track of the user's score and track how many

turns they have used. When the game ends we will present a summary to the user. Start by outputting a message welcoming the user and identifying yourself. The message must have your name, student number, and the phrase "COMP 10001 – W2020 Final Project". This message and your statement of authorship as a comment in the source code is required. A severe penalty will be applied if they fail to appear.

```
COMP 10001 - W2020 Final Project by Sample Solution, Student number 000123456
Welcome to my dice game, good luck!
```

The game is a simple dice game. If you wish to understand it better, try it with real dice and keep track of your score on paper.

At the start of a game the player will have a score of 0. They will tell you how many sides (also called faces) are on the die that they are using, at least 2 and at most 20. They will also indicate how many dice they intend to use, at least 3 and at most 6.

```
Enter # of faces [2,20]: -5
I'm sorry, that isn't a valid positive integer, please try again.
Enter # of faces [2,20]: seven
I'm sorry, that isn't a valid positive integer, please try again.
Enter # of faces [2,20]: 10
Enter # of dice [3,6]: some
I'm sorry, that isn't a valid positive integer, please try again.
Enter # of dice [3,6]: 10
I'm sorry, that isn't in the range 3 to 6 , please try again.
Enter # of dice [3,6]: 4
```

Figure 1: Sample program run 1

Once they have chosen valid values for these variables you will start playing a turn by rolling all of the dice. *HINT:* You should create a list that will hold your die rolls since you don't know how many dice will be used from game to game. Each element in your list represents a single die roll (see the hints section below for rolling a single die).

After the dice are rolled show the user which dice were rolled, the sum of the dice, and the average die roll – rounded to the nearest integer. Here are some examples.

```
You have rolled: [2, 1, 1]
These die sum to 4 and have an average rounded value of 1
```

Figure 2: Dice roll example 2+1+1=4, 4/3=1.333…

```
You have rolled: [2, 2, 1, 3, 2, 1]
These die sum to 11 and have an average rounded value of 2
```

Figure 3: Dice roll example 2+2+1+3+2+1 = 11, 11/6 = 1.83….

COMP 10001 Programming Fundamentals
Final Project – Winter 2020

We will also show the user the score that these dice are worth. Calculate the score by computing the percentage of a maximum score that they've earned times a bonus factor. Show this score as an integer – truncate any decimal portion.

To calculate the maximum score calculate the number of sides on each die * the number of dice. Their rolled sum / the maximum score is a percentage. Do not round the percentage, keep all decimal places.

For example:

Number of Sides	Number of Dice	Maximum Score	Rolled Sum	Percentage
6	5	6 * 5 = 30	12	12/30 = 0.4
6	5	6 * 5 = 30	19	19/30 = 0.633…
7	4	7 * 4 = 28	28	28/28 = 1.0

To calculate the bonus factor check if the dice rolled match any of the following patterns. If a pattern is matched show a message to the user. If a rule is not matched show a message explaining why. The messages should include relevant details used to make the decision. If they match more than 1 pattern sum all the matched bonus factors together. Be careful, not every pattern applies to every game. Finally add **(your student number mod 500)** to the score.

Pattern	Condition	Bonus Factor
all dice have the same value	# of sides >= 4	10
all dice add up to a prime number	maximum score >= 20	15
at least half the dice are greater than or equal to the rounded average	# of dice >= 5	5
all of the dice are different values	# of dice > 4 and # of sides > # of dice	8
no pattern matches	any game	1

You may use <u>any</u> method to determine if the dice total is a prime number, but it has to work for any possible value that might be encountered in the game (*HINT*: what is the absolute highest maximum score possible?)

Some examples:

Number of Sides	Number of Dice	Maximum Score	Roll	Sum	Avg	Bonus Factor	Patterns Matched
6	5	30	[3,4,2,1,5]	15	3	5+8	3,4
3	6	18	[2,2,2,2,2,1]	11	2	5	3
7	3	21	[7,1,3]	11	4	10+15+8	1,2,4

Which when implemented in code might look like:

```
Enter # of faces [2,20]: 7

Enter # of dice [3,6]: 4

You have rolled: [5, 4, 1, 1]

These die sum to 11 and have an average rounded value of 3
Pattern 1 not matched in your roll, [5, 4, 1, 1] ... some dice are different.
Pattern 2 matched, 11 is a prime number!
Pattern 3 does not apply since there are less than 5 dice.
Pattern 4 does not apply, either sides <= 4 or # sides <= # dice.
Since you matched a pattern, pattern 5 is not matched!
Your bonus factor is 15
These dice are worth 241 points.
```

Here we have a percentage of (11 / 28) = 0.3928571 … and our bonus factor is 15 for the prime number rule. 0.392…* 15 = 5.89... My student number % 2020 = 236. So adding it all up with rounding I get 241 points.

Another example:

```
Enter # of faces [2,20]: 6

Enter # of dice [3,6]: 5

You have rolled: [4, 6, 6, 6, 5]

These die sum to 27 and have an average rounded value of 5
Pattern 1 not matched in your roll, [4, 6, 6, 6, 5] ... some dice are different.
Pattern 2 not matched, 27 is not a prime number!
Pattern 3 matched! More than half of [4, 6, 6, 6, 5] are greater than or equal to the average of 5.
Pattern 4 not matched!  Some of the values in [4, 6, 6, 6, 5] match!
Since you matched a pattern, pattern 5 is not matched!
Your bonus factor is 5
These dice are worth 240 points.
```

Once the user has seen their score they have 1 chance to keep or reroll some or all of the dice. You can pick a technique for keeping track of the user's decisions on which die to reroll, but don't reroll them until they're done picking. Before you reroll ask them if they are sure. *Hint:* Since you don't know how many dice the user might pick, what about an array of Boolean values to tell you if you should reroll or not? If the 3rd Boolean is true, that means reroll the 3rd die, or something like that.

Though you may use other techniques, one simple way is to ask if the user wishes to reroll the dice one by one. For example: "Do you wish to reroll die 1?", "Do you wish to reroll die 2?" and so on. You might also ask them to enter the dice numbers to reroll all on one line and then use string manipulation to parse it. Remember to tell the user what the dice numbers are or remind them of the values somehow – don't assume they know arrays start counting at 0.

COMP 10001 Programming Fundamentals
Final Project – Winter 2020

Starting with `You have rolled: [4, 6, 6, 6, 5]`

The implementation might look like:

```
Do you want to reroll any dice? ['yes', 'no'] sure
I'm sorry, the choices are ['yes', 'no'] . Please pick one of them.
Do you want to reroll any dice? ['yes', 'no'] YES
Reroll die 1 (was 4) ['y', 'n'] fo shizzle
I'm sorry, the choices are ['y', 'n'] . Please pick one of them.
Reroll die 1 (was 4) ['y', 'n'] y
Reroll die 2 (was 6) ['y', 'n'] nyet
I'm sorry, the choices are ['y', 'n'] . Please pick one of them.
Reroll die 2 (was 6) ['y', 'n'] negatory
I'm sorry, the choices are ['y', 'n'] . Please pick one of them.
Reroll die 2 (was 6) ['y', 'n'] n
Reroll die 3 (was 6) ['y', 'n'] n
Reroll die 4 (was 6) ['y', 'n'] y
Reroll die 5 (was 5) ['y', 'n'] y
Are you sure? ['yes', 'no'] uh huh!
I'm sorry, the choices are ['yes', 'no'] . Please pick one of them.
Are you sure? ['yes', 'no'] YeS
```

If they say yes then reroll the dice and show the same output as the original roll with the new dice. If they say no then just move to the next step.

```
These die sum to 22 and have an average rounded value of 4
Pattern 1 not matched in your roll, [1, 4, 5, 6, 6] ... some dice are different.
Pattern 2 not matched, 22 is not a prime number!
Pattern 3 matched! More than half of [1, 4, 5, 6, 6] are greater than or equal to the average of 4.
Pattern 4 not matched!  Some of the values in [1, 4, 5, 6, 6] match!
Since you matched a pattern, pattern 5 is not matched!
Your bonus factor is 5
These dice are worth 239 points.
```

After they have either finished their roll or reroll the turn is over. Add whatever score they have earned onto a list of scores. If the user has played only 1 turn automatically start another turn.

```
These dice are worth 240 points.

This was your first turn, let's go again!

You have rolled: [1, 4, 5, 6, 6]
```

If this was their second or higher turn, tell the user whether or not the score from the last game was above average, average, or below average compared to the other games they have played. Then ask them if they would like to play another turn. Repeat the game as many times as the user likes, and print the total number of turns played and the average score when they choose to quit. The average should be shown as a rounded integer value.

COMP 10001 Programming Fundamentals
Final Project – Winter 2020

```
Do you want to reroll any dice? ['yes', 'no'] no
Your score of 125 on turn 2 was below average compared to other turns today.
Would you like to play another turn? ['y', 'n'] y

You have rolled: [1, 2, 1, 4, 2, 1]
```

[…]

```
Your bonus factor is 20
These dice are worth 129 points.

Do you want to reroll any dice? ['yes', 'no'] no
Your score of 129 on turn 3 was above average compared to other turns today.
Would you like to play another turn? ['y', 'n'] n
You played 3 turns today with an average score of 127 points
```

There is a full sample program run attached to the end of this document.

Hints

Simulating dice rolls

Rolling a die is really just a way to get a random number between 1 and the total number of faces on the die. A traditional cube die has 6 sides, when you roll it you get a number 1,2,3,4,5, or 6. We can simulate this by generating a random integer number using "**random.randint(1,6)**". See https://www.pythoncentral.io/how-to-generate-a-random-number-in-python/ for more help.

Rounding numbers

Remember the round() function from labs? It's format is **round(numberToRound, numberOfDecimalPlaces)**. For example **round(3.1415, 2)** gives back **3.14.**

(continued on next page)

COMP 10001 Programming Fundamentals
Final Project – Winter 2020

Deliverables and Evaluation Scheme

You will submit 2 files, unarchived to eLearn. **Any archived files will be awarded a grade of 0.**

File #1 – Microsoft Word Document

You are expected to provide the following evidence of a DDIT approach as taught in the course.

Requirement 1: IPO CHART 20%

Create an IPO chart for the main game loop using the techniques outlined in week 2's IPO chart and Pseudocode example guide and your lab submissions. Do NOT explain any of the methods that are defined in the starter code, for your pseudocode you can use them as if they made perfect sense. For example, you might say "RUN rollDie with userNumber" if userNumber was input earlier. Anything reasonable will be accepted but you do not have to explain how the modules work, only refer to them when creating your IPO chart.

Requirement 2: FLOW CHART 20%

Create a flow chart for the module specified in the below table. Turning in a flow chart for the wrong module is an automatic 0 for this section.

If your student number ends with:	Your flow chart is of module:
0,9	validateInt(min, max, prompt)
1,8	pattern1(sides, dice)
2,7	pattern2(sides, count, dice)
3,6	pattern3(dice)
4,5	pattern4(sides, dice)

For your flowcharts you may assume the input parameters are initialized, but be sure to state any assumptions about these values.

Requirement 3: TESTING 10%

Demonstrate complete coverage traces of your flow chart. Pick your values deliberately to ensure that every path in your flow chart is followed and produces the correct answer.

COMP 10001 Programming Fundamentals
Final Project – Winter 2020

File #2 – Python program source code 50%

Implement the program described here in Python. Use the provided starter code as a basis for your work. You are required to use elements from all aspects of the course.

You will be evaluated for

- Use of appropriate syntactic structures (sequential, selection, repetition, modules, lists)
- Appropriate choice of data types (at least 3 of the 4 data types are required)
- Variable names, comments, etc (general readability)
- Correctness – does the program work?

The marking standard will follow the same guidelines as demonstrated through the labs this semester.

No other files will be accepted.

Please see the rubrics attached to understand the evaluation structure.

Any questions about this assignment should be directed to your professor more than 24 hours prior to the due date.

COMP 10001 Programming Fundamentals
Final Project – Winter 2020

Rubrics

IPO Chart	Meets or Exceeds Expectations (> 5)	Approaches Expectation (3-5)	Below Expectations (< 3)
Coverage 7 marks	All inputs and outputs are specified. Processing steps are outlined in sufficient detail to clearly understand without further explanation.	Most or all input and outputs are specified. Processing steps require additional refinement to clarify the exact actions required.	Many inputs or outputs are not identified. Processing steps are not granular and the steps required for implementation are unclear.
Algorithm 7 marks	Steps are ordered logically and follow from one another. Back checking does not expose any variables that "appear from nowhere".	Some steps are out of order and/or back checking demonstrates that some variables are used before being defined.	Steps lack an overall order as a constructed process that leads from inputs to outputs.
Pseudocode 6 marks	Pseudocode is specific enough to be understood precisely, but is general enough to be adapted to any programming language, and is not simply Python code.	Pseudocode is fairly specific with perhaps some ambiguities. Portions may be too specific, or resemble Python programming too closely.	Pseudocode is both highly ambiguous and lacking detail or is essentially Python code.

COMP 10001 Programming Fundamentals
Final Project – Winter 2020

Flow Chart a/b c	Meets or Exceeds Expectations (4-5) (> 7)	Approaches Expectation (2-3) (4-7)	Below Expectations (< 2) (<4)
Organization **5 marks**	Flow chart is neat and traversal flows logically from the start to the finish along all paths.	Flow chart is fairly neat and traversal flows from the start to the finish with some issues.	Flow chart is disorganized and difficult to follow. Traversal from start to finish is unclear in some sections.
Symbols & Connectedness **5 marks**	Uses established symbols for the course correctly. Circles, rectangles, and diamonds have the correct number of arrows entering and exiting from them.	Mostly uses the established symbols for the course correctly. Symbols generally have the correct number of arrows entering and exiting them.	Symbols are frequently not used correctly and/or the number of arrows entering and exiting symbols is not consistently correct.
Algorithm **10 marks**	Algorithm demonstrates a correct solution to the problem posed.	Algorithm demonstrates a solution to the problem posed with some minor issues or exceptions.	Algorithm does not demonstrate a correct solution to the problem posed.

Testing	Meets or Exceeds Expectations (9-10)	Approaches Expectation (5-8)	Below Expectations (< 3)
Coverage **10 marks**	Traces choose variables carefully to ensure near minimal amount of testing that achieves complete coverage of the flow chart along all branches.	Traces do not choose variables optimally resulting in extra traces however, the resulting work achieves complete coverage along all branches.	Testing does not demonstrate complete coverage.
Programming	Meets or Exceeds Expectations	Approaches Expectation	Below Expectations

COMP 10001 Programming Fundamentals
Final Project – Winter 2020

Performance **20 marks**	Program implements most features of the specification correctly and with care provided to input validation. **(15-20)**	Program implements many features of the specification. Omissions are not severe in nature and the overall feel of the program is still present. **(10-14)**	Program does not run and must be manually inspected to evaluate completion or fails to substantially implement the requested solution. **(< 5)**
Structure **20 marks**	The programmer chooses structures that are appropriate to the creation of the solution. This includes using lists where possible. If structures are logical and make use of compound Boolean expressions where appropriate. Data types are chosen with appropriately. **(> 16)**	The programmer generally has chosen structures appropriate to the creation of the solution. Some topics may be omitted, such as failing to use lists. If structures while functional do not use compound Boolean expressions. Data types are often chosen appropriately. **(10 – 15)**	The programmer has not demonstrated care with the selection of structures. Many topics from the course are omitted including lists, string manipulation, compound Boolean expressions, etc. Data types may be poorly chosen. **(< 5)**
Readability **10 marks**	Programmer has commented the code appropriately to make clear program statements. Variable names are generally well chosen and appropriate. **(7-10)**	The code is generally well commented but may have areas that are less so. Variable names may be frequently meaningless or unclear to the reader. **(4-7)**	The code lacks comments overall and/or variable names are frequently unclear or meaningless. **(1-2)**

COMP 10001 Programming Fundamentals
Final Project – Winter 2020

Sample program execution

This is an example of a full run of the program. Your output does not have to match it exactly but it should behave the same in all substantial ways. I have highlighted the user input in red for clarity, you are not required to change colours.

COMP 10001 - W2020 Final Project by Sample Solution, Student number 000123456
Welcome to my dice game, good luck!

Enter # of faces [2,20]: 10

Enter # of dice [3,6]: 4

You have rolled: [9, 3, 8, 8]

These die sum to 28 and have an average rounded value of 7
Pattern 1 not matched in your roll, [9, 3, 8, 8] ... some dice are different.
Pattern 2 not matched, 28 is not a prime number!
Pattern 3 does not apply since there are less than 5 dice.
Pattern 4 does not apply, either sides <= 4 or # sides <= # dice.
Since none of the other patterns were matched, pattern 5 is matched!
Your bonus factor is 1
These dice are worth 236 points.

Do you want to reroll any dice? ['yes', 'no'] YeS
Reroll die 1 (was 9) ['y', 'n'] n
Reroll die 2 (was 3) ['y', 'n'] yep
I'm sorry, the choices are ['y', 'n']. Please pick one of them.
Reroll die 2 (was 3) ['y', 'n'] y
Reroll die 3 (was 8) ['y', 'n'] N
Reroll die 4 (was 8) ['y', 'n'] n
Are you sure? ['yes', 'no'] Yup
I'm sorry, the choices are ['yes', 'no']. Please pick one of them.
Are you sure? ['yes', 'no'] YeS

You have rolled: [9, 2, 8, 8]

These die sum to 27 and have an average rounded value of 7
Pattern 1 not matched in your roll, [9, 2, 8, 8] ... some dice are different.
Pattern 2 not matched, 27 is not a prime number!
Pattern 3 does not apply since there are less than 5 dice.
Pattern 4 does not apply, either sides <= 4 or # sides <= # dice.
Since none of the other patterns were matched, pattern 5 is matched!
Your bonus factor is 1
These dice are worth 236 points.

COMP 10001 Programming Fundamentals
Final Project – Winter 2020

This was your first turn, let's go again!

You have rolled: [5, 10, 1, 5]

These die sum to 21 and have an average rounded value of 5
Pattern 1 not matched in your roll, [5, 10, 1, 5] ... some dice are different.
Pattern 2 not matched, 21 is not a prime number!
Pattern 3 does not apply since there are less than 5 dice.
Pattern 4 does not apply, either sides <= 4 or # sides <= # dice.
Since none of the other patterns were matched, pattern 5 is matched!
Your bonus factor is 1
These dice are worth 236 points.

Do you want to reroll any dice? ['yes', 'no'] no
Your score of 236 on turn 2 was average compared to other turns today.
Would you like to play another turn? ['y', 'n'] yes
I'm sorry, the choices are ['y', 'n']. Please pick one of them.
Would you like to play another turn? ['y', 'n'] y

You have rolled: [7, 2, 6, 1]

These die sum to 16 and have an average rounded value of 4
Pattern 1 not matched in your roll, [7, 2, 6, 1] ... some dice are different.
Pattern 2 not matched, 16 is not a prime number!
Pattern 3 does not apply since there are less than 5 dice.
Pattern 4 does not apply, either sides <= 4 or # sides <= # dice.
Since none of the other patterns were matched, pattern 5 is matched!
Your bonus factor is 1
These dice are worth 236 points.

Do you want to reroll any dice? ['yes', 'no'] no
Your score of 236 on turn 3 was average compared to other turns today.
Would you like to play another turn? ['y', 'n'] n
You played 3 turns today with an average score of 236 points

=== END ===

