



ICT 133

Structured Programming

Tutor-Marked Assignment

July 2019 Presentation

TUTOR-MARKED ASSIGNMENT (TMA)

This assignment is worth 18 % of the final mark for ICT133, Structured Programming.

The cut-off date for this assignment is **Wednesday 04 September 2019, 2355hrs.**

Assignment Requirements:

Do NOT define classes for this TMA.

Unless specified in the question, you CANNOT use packages not covered in this module, e.g., numpy, pandas etc..

Provide sufficient comments to your code, and ensure that your program adheres to good programming practices such as not using global variables.

Failing to do so can incur a penalty of as much as 50% of the mark allotted.

Submission Details:

Use the template word document provided - **SUSS_PI_No-FullName_TMA.docx**. Rename the file with your SUSS PI and full name join with “_TMA” e.g. “**SUSS_PI-TomTanKinMeng_TMA.docx**” (without the quotes).

Include the following particulars on the first page of the word document, on separate lines: **Course Code, SUSS PI No., Your Name, Tutorial Group and Submission Date.**

Copy and paste the source code of each program you write in **text** format. Submit screenshots for **only** output of your program, where applicable.

If you submit the source code as a screenshot, your code will not be marked. That is, **screenshot code will be awarded zero mark.**

Submit your solution in the form of a **single MS Word document. Do NOT submit as a pdf document.** You will be penalised if you fail to submit a word document.

You should make only one submission for TMA.

Question 1

Write complete full Python programs for each part, to perform the following tasks

- a) Write a Python program to perform the following tasks.
- Prompt a user to enter the following pieces of information:
 - the initial value of an investment, i
 - the expected final value of an investment, f and
 - the number of years to accumulate the initial value to reach the final value, n .
 - Compute the annual compound interest rate in percent, r using the formula:

$$r = 100\left(\sqrt[n]{\frac{f}{i}} - 1\right)$$

- Prompt the user to enter the number of years, y , he wishes to invest.
- Compute the final amount based on the computed compound interest rate, using the formula:

$$f = i\left(1 + \frac{r}{100}\right)^y$$
- Display the final amount based on the computed compound interest rate. Numbers should be displayed with 2 decimal digits.

An example run with user input in green is shown here:

```
Enter the initial bank balance: 100
Enter the final bank balance: 120
Enter the number of years to reach final balance: 5
How many years to deposit given average interest rate is 3.71%: 10
After 10 years, you will have $144.00
```

(3 marks)

- b) Make a copy of your Python program in part a) for this part as you need to submit the program for part a).
- Add a check that if the final amount is less than the initial amount, prompt whether the user wishes to swap the values of the initial amount with the final amount. If the user does not wish to swap, proceed as usual. Allow the user to respond in uppercase or lowercase. You may assume that the user enters either an n or a y as the first non-whitespace character.

Two sample runs with user input in green are shown here:

Run 1:

```

Enter the initial bank balance: 200
Enter the final bank balance: 100
Initial value: $200 is more than final value: $100
Do you wish to swap the values? (y/n): n
OK. Getting negative returns
Enter the number of years to reach final balance: 5
How many years to deposit given average interest rate is -12.94%:
10
After 10 years, you will have $50.00

```

Run 2:

```

Enter the initial bank balance: 200
Enter the final bank balance: 100
Initial value: $200 is more than final value: $100
Do you wish to swap the values? (y/n): yes
The values have been swapped
Enter the number of years to reach final balance: 5
How many years to deposit given average interest rate is 14.87%: 10
After 10 years, you will have $400.00

```

(3 marks)

- c) Make a copy of your Python program in part b) for this part as you need to submit the program for part b).
 Add a check to ensure that the user enters either an n or a y as the first non-whitespace character when prompted whether he wishes to swap. You should repeatedly prompt the user until he enters a valid value.

An example run with user input in green is shown here:

```

Enter the initial bank balance: 200
Enter the final bank balance: 100
Initial value: $200 is more than final value: $100
Do you wish to swap the values? (y/n):
Do you wish to swap the values? (y/n): an
Do you wish to swap the values? (y/n): a
Do you wish to swap the values? (y/n): na
OK. Getting negative returns
Enter the number of years to reach final balance: 5
How many years to deposit given average interest rate is -12.94%:
10
After 10 years, you will have $50.00

```

(3 marks)

- d) Make a copy of your Python program in part c) for this part as you need to submit the program for part c).

Modify the code segment for part a) so that the program display the yearly value of the investment amount from the initial amount to the final amount, instead of displaying only the final amount.

An example run with user input in green is shown here:

```

Enter the initial bank balance: 200
Enter the final bank balance: 100
Initial value: $200 is more than final value: $100
Do you wish to swap the values? (y/n): nah
OK. Getting negative returns
Enter the number of years to reach final balance: 5
How many years to deposit given average interest rate is -12.94%:
10
Year Amount($)
1    174.11
2    151.57
3    131.95
4    114.87
5    100.00
6     87.06
7     75.79
8     65.98
9     57.43
10    50.00

```

(3 marks)

- e) Make a copy of your Python program in part d) for this part as you need to submit the program for part d).

Allow the user to repeatedly enter new initial and final values until he enters 0 for the initial amount. Perform the steps as usual for each pair of initial and final values. You must modularise your program for this part by defining at least 2 functions.

An example run with user input in green is shown here:

```

Enter the initial bank balance: 100
Enter the final bank balance: 200
Enter the number of years to reach final balance: 2
How many years to deposit given average interest rate is 41.42%: 5
Year Amount($)
1    141.42
2    200.00
3    282.84
4    400.00
5    565.69

```

```

Enter the initial bank balance: 500
Enter the final bank balance: 300
Initial value: $500 is more than final value: $300
Do you wish to swap the values? (y/n): n
OK. Getting negative returns
Enter the number of years to reach final balance: 2
How many years to deposit given average interest rate is -22.54%: 5
Year Amount($)
1      387.30
2      300.00
3      232.38
4      180.00
5      139.43
Enter the initial bank balance: 0
Program end

```

(8 marks)

Paste the whole Python programs for each part: a) to e) as text into the word document. You should submit 5 complete programs. Screenshot output obtained from the execution of each program, and paste them into the word document.

Question 2

To enable collaborative learning through participation and engagement, you are encouraged to make postings in a discussion forum. You must NOT post Python code in the discussion forum.

The problem:

An expression is something that evaluates to a value, e.g., $3 + 4$ evaluates to 7. As you learn to write Python programs, you will realise that writing syntactically correct expressions is a must for them to be evaluated by the Python interpreter. For example, $3 + / 4$ is not a valid expression, and so, the Python interpreter cannot evaluate it.

In this question, you are to solve the problem of determining whether an expression is valid, and to evaluate it if it is so. Otherwise, you are to provide a helpful error message. In other words, you are to write a program to handle a small portion of what a Python interpreter needs to do when it gets an expression to evaluate.

Consider only simplified expressions as described here:

- An expression may contain only these binary operators: $+$, $-$, $/$, $//$ and $*$.
- Parentheses, that is, (and), can be used to change the order of the evaluation of the operations.
- Operands are negative or positive numeric values (int or float).
- Operands cannot be variables.

The Python interpreter needs to tokenise the expression, however, you can simplify the tokenising process by assuming that the tokens (number, parentheses and operators) in the expression will be separated by whitespace. Thus, you will be able to get a next token by moving past whitespaces.

Examples of invalid expressions with explanation are given below:

Invalid expressions	Explanation
$1 + 2 + a + 4$	Operand must be a number, a is not a number
$(1 + 3))$	Unbalanced parentheses
$(1 + 3) * ((2)$	Unbalanced parentheses
$33.3 + + 1$	Operators require a left and right numeric operands
$(33) (1)$	Operands must be separated by an operator
$) ($	No preceding left parenthesis for the right parenthesis
$()$	No operand within the bracket

Examples of valid expressions with explanation and the evaluated result are given below:

valid expressions	Reasons	Result of evaluation
$1.2 + -3$	Operand operator operand	-1.8
$((1 + 3))$	Balanced parentheses, operand operator operand	4
$(1 + 3) * ((2))$	Balanced parentheses, operand operator operand	8
$33.3 + -2 + 1$	Operand operator operand	32.3
$(33) / (-1)$	Operand operator operand	-33.0
$(33) // (1)$	Operand operator operand	33

a) Before embarking on the program development, journal your learning journey by making postings for ALL items listed below (You must NOT POST ANY PYTHON CODE, and your posting should be clear, understandable and with SUFFICIENT elaboration):

- at least TWO difficulties or questions you have about the problem
- at least TWO difficulties or questions you have when attempting to solve the problem,
- at least TWO suggestions, clarifications, observation or feedback to other students' postings, accompanied with reasons
- at least TWO conclusions of your research on the problem and how to solve it (Note that no Python code should be posted)
- at least TWO important lessons you have through this exercise.

Copy and paste at least THREE (3) postings that collectively cover ALL FIVE (5) bulleted items. The postings you submit should have at least THREE (3) different posting dates on THREE (3) different weeks. Each posting should not cover more than 2 of the 5 bulleted items. If postings are made on the same week (Sun-Sat), only one of them will be awarded mark.

(12 marks)

b) Demonstrate your understanding of modular program design, and develop the program that repeatedly read an expression and output only one of the following messages:

- Unexpected right parenthesis between *expr_left* and *)expr_right*
- There is 1 unclosed left parenthesis in *expr*

- There are n unclosed left parentheses in *expr*
- Invalid expression, expecting operator between *expr_left* and *expr_right*
- Invalid expression, expecting operand between *expr_left* and *expr_right*
- Invalid expression, expecting operand between *expr_left* and *expr_right* but *currentToken* is a variable
- result from evaluating the expression.

You may use the function `eval` to evaluate an expression.

Note that *expr_left* refers to the portion of the expression without error and *expr_right* refers to start of where the error occurs in the expression. *expr* refers to the input expression, and *currentToken* refers to the current token being processed.

Your program should process the tokens in the input expressions without generating exceptions, and your program cannot use the `try-except` construct.

An example run with user input in green is shown here:

```

Enter an expression: 1 + 2 + a + 4
Invalid expression, expecting operand between 1 + 2 + and + 4 but a
is a variable
Enter an expression: ( 1 + 3 ) )
Unexpected right parenthesis between ( 1 + 3 ) and )
Enter an expression: ( 1 + 3 ) * ( ( 2 )
There is 1 unclosed left parenthesis in ( 1 + 3 ) * ( ( 2 )
Enter an expression: 33.3 + + 1
Invalid expression, expecting operand between 33.3 + and + 1
Enter an expression: ( 33 ) ( 1 )
Invalid expression, expecting operator between ( 33 ) and ( 1 )
Enter an expression: ) (
Unexpected right parenthesis between and ) (
Enter an expression: ( )
Invalid expression, expecting operand between ( and )
Enter an expression: 1.2 + -3
The answer is -1.8
Enter an expression: ( ( 1 + 3 ) )
The answer is 4
Enter an expression: ( 1 + 3 ) * ( ( 2 ) )
The answer is 8
Enter an expression: 33.3 + -2 + 1
The answer is 32.3
Enter an expression: ( 33 ) / ( -1 )
Invalid expression, expecting operand between ( and 33 ) / ( -1 )
Enter an expression: ( 33 ) / ( -1 )
The answer is -33.0
Enter an expression: ( 33 ) // ( 1 )
The answer is 33
Enter an expression:
Application ended

```

(18 marks)

Screenshot your postings for part Q2a), and paste them into the word document. Ensure that your screenshots are readable. Copy also the links to each of your postings.

Paste the Python code for part Q2b) as text into the word document. Screenshot output obtained from the execution of the program, and paste them into the word document.

Question 3

You are required to develop an application to manage a patient queue in a clinic. Details recorded for each patient in a wait queue are as follow:

- patient id
- patient name
- patient queue number
The queue number is a running number starting from 1.
The first registered patient has queue number 1, the second registered patient has queue number 2, the third registered patient has queue number 3, and so on.
- number of times the patient's queue number is called but he missed the calls
When a patient is called, the probability that he is ready to see the doctor is 25%. That is, assume that 75% of the time, the patient misses seeing the doctor when his queue number is called. When the patient misses seeing the doctor, the number of call he misses is increased by 1.

The application allows the clinic staff to select tasks from the following menu:

Menu

1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit

Apply modular programming and decide on the functions to implement the application. **You must not use the collection type, *dict* for this question.** Add any other steps not described for the options and for the main function of the application to fulfil their requirements:

- Register a Patient
 - Prompt the user for the patient id.
 - If the patient id is already in the wait queue, display that patient is already in the wait queue, together with his queue number and his position in the queue.
 - Otherwise, prompt for the patient's name, and add him into the wait queue with the details: patient id, name, the queue number given at registration, and the number of calls missed set to 0 initially.
 - Add the newly registered patient to the wait queue.
 - Display the patient id and his registered queue number.

(7 marks)

- **Call Next Patient**
 - Display that there is no patient in wait queue if the wait queue is empty.
 - Otherwise, display the queue number of the first patient in the wait queue.
 - In the event that the patient is ready to see the doctor, he is removed from the wait queue, and put into another 'seen doctor' queue, to indicate that the patient has gotten to see doctor. Display a message to indicate that the patient is being served.
 - In the event that the patient misses seeing the doctor when his queue number is called, add 1 to the number of call he misses.
 - If the number of calls missed becomes 3, the patient is put to the end of the wait queue. Reset his number of calls missed to zero. Display a message to indicate that the patient has been pushed to the end of the wait queue. The next patient in the wait queue is called, and the same process is repeated until one patient responded to the call or until the end of the wait queue is reached.
 - If end of the wait queue is reached and none of the patients responded, display the message that none of the patients in the queue is responding.
(7 marks)

- **List Patients in Queue**
 - Display that there is no patient in wait queue if the wait queue is empty
 - Display a header for the patient detail: patient id, name, queue number, number of missed calls.
 - Display the details of patients in wait queue. Ensure that the details show up in the correct header.
(2 marks)

- **Search Patient's Queue Position**
 - Display that there is no patient in wait queue if the wait queue is empty
 - Otherwise, prompt for patient id
 - Display patient is not in the wait queue if the patient id cannot be located in queue
 - Display the patient's id, name, queue number and position in wait queue if the patient id can be located in queue.
(3 marks)

- **Record Patients Seen**
 - Append the details of patients in the 'seen doctor' queue: patient id, name and queue number to a file, `seenQ3.txt`.
 - Empty the 'seen doctor' queue. Note that the queue may (re)grow when option **Call Next Patient** is called.
 - Display the number of patient details written into the file.
(3 marks)

- **Exit**

If there are patients in the 'seen doctor' queue but their details have not been written to `seenQ3.txt` yet, then append them to the file before ending the application. Note that the details include patient id, name and queue number.
(1 mark)

Write a `main` function to repeatedly display the menu to allow a clinic staff (the user) to select an option to call the appropriate functions to perform task according to the specifications. The application ends when the user enters 0 to exit.

(2 marks)

An example run with user input in green is shown here:

```
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 2
No patient in queue
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 3
No patient in Queue
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 4
No patient in Queue
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 5
0 patient added
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 1
Enter patient's id: T123
Enter patient's name: Tom
T123 is registered with queue number 1
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 1
Enter patient's id: T098
Enter patient's name: Jane
T098 is registered with queue number 2
```

```

Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 1
Enter patient's id: S222
Enter patient's name: Alice
S222 is registered with queue number 3
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 3
PatientId Name Queue Number Calls missed
T123 Tom 1 0
T098 Jane 2 0
S222 Alice 3 0
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 1
Enter patient's id: T098
Patient id T098 is in position 2
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 2
Calling queue number 1
Calling queue number 2
Calling queue number 3
No patient is responding
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 3
PatientId Name Queue Number Calls missed
T123 Tom 1 1
T098 Jane 2 1
S222 Alice 3 1
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 2
Calling queue number 1
Serving queue number 1

```

```

Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 4
Enter patient's id: T123
Patient id T123 is not in queue
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 4
Enter patient's id: T098
Patient id T098 is in position 1
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 2
Calling queue number 2
Calling queue number 3
No patient is responding
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 3
PatientId  Name           Queue Number  Calls missed
T098      Jane                2             2
S222      Alice               3             2
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 2
Calling queue number 2
Serving queue number 2
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 5
2 patients added
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen

```

```

0. Exit
Enter option: 2
Calling queue number 3
S222 is moved to end of patient queue
No patient is responding
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 2
Calling queue number 3
No patient is responding
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 1
Enter patient's id: T999
Enter patient's name: Peter
T999 is registered with queue number 4
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 3
PatientId  Name           Queue Number  Calls missed
S222      Alice           3             1
T999      Peter           4             0
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 2
Calling queue number 3
S222 is moved to end of patient queue
Calling queue number 4
Serving queue number 4
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 5
1 patient added
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 2
Calling queue number 3

```

```

Serving queue number 3
Menu
1. Register a Patient
2. Call Next Patient
3. List Patients in Queue
4. Search Patient's Queue Position
5. Record Patients Seen
0. Exit
Enter option: 0
1 patient added
Application is closing

```

Content of file, seenQ3.txt which records details of patients who have seen doctor:

T123	Tom	1
T098	Jane	2
T999	Peter	4
S222	Alice	3

Paste the Python program as text into the word document. Screenshot output obtained from the execution of the program, and paste them into the word document.

Question 4

You are required to implement a simple variant of Blackjack using dice. The game is played among several players, and the winner is the player who has the highest total dice value.

Your program must keep track of players' name, points and dice values. You must **use the collection type *dict*** to store the players' details as well as the collection of players. You may use *list* for any other collection.

Refer to the example runs. Your program must adhere to these guidelines:

- There must be a minimum of 3 players and a maximum of 5.
Start by getting the names of 3 to 5 players. Players' names must be unique. (5 marks)
- Each player starts with 2 points (2 marks)
- For each new game, randomly order the players, so that they roll the dice according to this order. (3 marks)
- Each game starts with each player rolling 2 dice.
Display the total for each player. (5 marks)
- Each player then takes turn to decide whether to roll a dice or to miss.
If he rolls a dice, he can decide to keep rolling as long as the total is not 13 or more. Display the total for the player each time he rolls a dice. (5 marks)

- At the end of the game, the points for the players are adjusted as follows:
 - Add 2 points to players with the highest total dice value that does not exceed 13. There can be more than one winner.
 - Deduct 2 points from players who bust, that is, their total dice value exceeds 13.
 - Do not adjust the points for all other players.
 - Display the game summary.

(6 marks)

- The players can continue to play more games with their existing points as long as there is no player with 0 point or with 10 points.

(4 marks)

Two example runs are shown below:

Run 1:

```

Enter player name, <Enter> to end:
Minimum 3 players please, currently only 0 player!
Enter player name, <Enter> to end: A
Enter player name, <Enter> to end: B
Enter player name, <Enter> to end: C
Enter player name, <Enter> to end: D
Enter player name, <Enter> to end: E
5 players playing
Playing in this order: (D, E, A, B, C)
Start game. Roll 2 dice per player
D, dice: (2, 6), total 8
E, dice: (4, 1), total 5
A, dice: (6, 6), total 12
B, dice: (2, 6), total 8
C, dice: (5, 1), total 6
D, roll or miss? (r/m)? r
D, dice: (2, 6, 4), total 12
D, roll or miss? (r/m)? m
E, roll or miss? (r/m)? r
E, dice: (4, 1, 4), total 9
E, roll or miss? (r/m)? r
E, dice: (4, 1, 4, 5), total 14. Bust!
A, roll or miss? (r/m)? m
B, roll or miss? (r/m)? m
C, roll or miss? (r/m)? m
Game summary:
D, dice: (2, 6, 4), total 12. Winner! Points: 2+2 = 4
E, dice: (4, 1, 4, 5), total 14. Bust! Points: 2-2 = 0
A, dice: (6, 6), total 12. Winner! Points: 2+2 = 4
B, dice: (2, 6), total 8. Points: 2
C, dice: (5, 1), total 6. Points: 2
Player bust. End game

```

Run 2:

```

Enter player name, <Enter> to end: A
Enter player name, <Enter> to end: B
Enter player name, <Enter> to end: C
Enter player name, <Enter> to end:
3 players playing
Playing in this order: (B, C, A)

```

```

Start game. Roll 2 dice per player
B, dice: (1, 6), total 7
C, dice: (2, 2), total 4
A, dice: (1, 2), total 3
B, roll or miss? (r/m)? r
B, dice: (1, 6, 4), total 11
B, roll or miss? (r/m)? m
C, roll or miss? (r/m)? r
C, dice: (2, 2, 6), total 10
C, roll or miss? (r/m)? r
C, dice: (2, 2, 6, 3), total 13
A, roll or miss? (r/m)? r
A, dice: (1, 2, 4), total 7
A, roll or miss? (r/m)? r
A, dice: (1, 2, 4, 4), total 11
A, roll or miss? (r/m)? m
Game summary:
B, dice: (1, 6, 4), total 11. Points: 2
C, dice: (2, 2, 6, 3), total 13. Winner! Points: 2+2 = 4
A, dice: (1, 2, 4, 4), total 11. Points: 2
Continue next game (y/n)? y
Playing in this order: (C, A, B)
Start game. Roll 2 dice per player
C, dice: (1, 5), total 6
A, dice: (3, 5), total 8
B, dice: (2, 3), total 5
C, roll or miss? (r/m)? r
C, dice: (1, 5, 1), total 7
C, roll or miss? (r/m)? r
C, dice: (1, 5, 1, 1), total 8
C, roll or miss? (r/m)? m
A, roll or miss? (r/m)? m
B, roll or miss? (r/m)? r
B, dice: (2, 3, 1), total 6
B, roll or miss? (r/m)? r
B, dice: (2, 3, 1, 6), total 12
B, roll or miss? (r/m)? m
Game summary:
C, dice: (1, 5, 1, 1), total 8. Points: 4
A, dice: (3, 5), total 8. Points: 2
B, dice: (2, 3, 1, 6), total 12. Winner! Points: 2+2 = 4
Continue next game (y/n)? n
End game

```

Paste the Python program as text into the word document. Screenshot output obtained from the execution of the program for 2 different game sessions and paste them into the word document.

---- END OF ASSIGNMENT ----