SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

# ICT133
# Structured Programming

---

## Tutor-Marked Assignment

## July 2021 Presentation

---

## TUTOR-MARKED ASSIGNMENT (TMA)

This assignment is worth 24 % of the final mark for ICT133, Structured Programming

The cut-off date for this assignment is **Wednesday, 08 September 2021, 2355 hours.**

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. re, collections, numpy, pandas etc.**
- All functions must be documented. Provide sufficient comments to your code, and ensure that your program adheres to good programming practices such as not using global variables.
- **Do not use the exit() function.**
- 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.

*Answer all questions. (Total 100 marks)*

**Question 1 (25 marks)**

This program demonstrates operations of bitwise operators using binary digits. The bitwise operators are as follows:

| Symbol | Operation | Explanation |
|---|---|---|
| & | and | Sets result bit to 1 if both corresponding bits are 1.<br>E.g. 1010 & 1100 is 1000 |
| \| | or | Sets result bit to 1 if one of the two corresponding bits is 1.<br>E.g 1010 & 1100 is 1110 |
| ^ | xor | Sets result bit to 1 only if one of the corresponding bits is 1.<br>E.g. 1010 & 1100 is 0110 |

(a) Write and express a program that reads in a binary expression as a string and evaluates the result. Sample program executions are as follows:
Sample 1:

```
Enter binary expression: 110110 & 110011
Result: 110010
```

Sample 2:

```
Enter binary expression: 111 ^ 101
Result: 010
```

Assume that input string is valid, i.e. binary digits are of the same length and bitwise operator is valid. The expression has operands and operator separated by blanks in between. You can use the logical `and` `or` operators, but not allowed to use the Python bitwise operators in computing the results.

This program has only the `main()` function. **Do not write any other functions** for this program.

Paste screenshots of at least **TWO (2)** program executions, with different binary operators.

(8 marks)

(b) Write a function `validExpression(expression)` that has a string parameter representing a binary expression. Assume the string expression will be in the format of "`operand operator operand`", separated by blanks. These are the possible errors that may occur:

- Operand does not consist of binary digits
- Operands are of different length
- <operator> is invalid. Must be |, & or ^, where <operator> is the invalid operator

- Length of the operands must be at least 2 digits

If there is an error, the function prints the error and returns False. Otherwise, it returns True.

(8 marks)

(c) Write a `main()` function that can repeatedly enter binary expressions and prints the results. The program can keep on getting input and displaying result, until the `<Enter>` key is entered to end input. A sample program execution is as follows:

```
Enter binary expression: 121 & 110
Operand does not consist of binary digits
Enter binary expression: 110 ^ 101
  110
^ 101
-----
  011
Enter binary expression: 110 & 1110
The operands are of different length
Enter binary expression: 1 ^ 1
Length of the operands must be at least 2 digits
Enter binary expression: 110 * 101
* is invalid. Must be !, & or ^
Enter binary expression: 110110 & 101010
  110110
& 101010
--------
  100010
Enter binary expression: <Enter>
End of program.
```

Make use of the function in part (b) and add additional functions to make the program modular. The format of the output must be adhered to. Include a screenshot which shows a program execution that includes different length binary digits and validation errors.

(9 marks)

**Question 2 (20 marks)**

Develop and write a program that plays a computer letter 'shooting' game. A player generates a target which consists of a series of letters and the computer program attempts to 'shoot' the letters. The game ends when the program runs out of 'ammunition'.

The rules of the game are as follows:

- Player keys in a minimum 4 up to 12 unique letters to form the target. All letters will be converted to uppercase. Valid letters are from A-Z.
- Program will generate 'ammunition' letters. To have a 50% chance of 'hitting' a target letter, the letters formed for the "ammunition" will include the target string plus other unique letters to form double the length of the target. For example, if the target string is 'ABCD', then the ammunition may be 'ABCDUXWF'.
- The program picks a random letter from the 'ammunition' letters and 'shoots' at the target. If there is a hit, record the hit.
- The number of chances to shoot will be the length of the target string.
- At the end of the program, display the hits and misses.
- Program runs for only 1 game.

A sample program execution is as follows:

| | Comments |
|---|---|
| Enter target letters: **aa**<br>Must be unique letters unique and of length 4 to 12<br>Enter target letters: **a1234**<br>Must be unique letters unique and of length 4 to 12<br>Enter target letters: **ABCDEE**<br>Must be unique letters unique and of length 4 to 12<br>Enter target letters: **ABCD**<br>Ammunition ABCDNMRZ<br>Target: ABCD<br>Hit 1. Enter to shoot: **<Enter>**<br>letter D hit!<br>Target: ABC*<br>Hit 2. Enter to shoot: **<Enter>**<br>letter Z miss!<br>Target: ABC*<br>Hit 3. Enter to shoot: **<Enter>**<br>letter R miss!<br>Target: ABC*<br>Hit 4. Enter to shoot: **<Enter>**<br>letter A hit!<br>Target: *BC*<br>2 hits, 2 letters left! | Cheatsheet displays the "Ammunition letters".<br>Hit <Enter> key to simulate computer shoot. Letter is randomly picked from ammunition.<br>Display hit or miss.<br><br>No. of tries = length of target. Show a summary of hits and miss. If all the letters in the target are hit, display 'All hit!' instead |

(a) Write a function that inputs a string and checks that it has a minimum 4 up to 12 unique letters from A-Z. Display an error message if input is invalid and repeat the process. The function returns the target string in uppercase.

(6 marks)

(b) Write the program to play the game. Include a screenshot of a program execution.

(14 marks)

No data structures like lists or dictionary should be used for this question. Apply only strings, string methods, control structures and functions are sufficient. Keep the program modular by making use of functions.

**Question 3 (30 marks)**

This is a program that challenges how fast a user can add digits of a number.

(a) Write and develop a program that allows a user 3 tries to sum numbers of a fixed length of 4. Use only digits 1 to 9 for the number. The time taken in seconds for the user to input the sum is captured. After 3 tries, a score summary is displayed. A sample program execution is as follows:

| | |
|---|---|
| Digits: 2992<br>Try 1, sum the digits: **22**<br>Digits: 5238<br>Try 2, sum the digits: **<Enter>**<br>Try 2, sum the digits: **18**<br>Digits: 5485<br>Try 3, sum the digits: **21**<br>Summary<br>Try 1: 2992 22 correct 6.6 secs<br>Try 2: 5238 18 correct 6.8 secs<br>Try 3: 5485 21 incorrect --.- secs<br>2/3 correct. Average time is 6.7 secs | Comments<br>Do not include 0. Only 1 to 9.<br><br>Continue to prompt for input if <Enter> key.<br><br>Incorrect total, no timing.<br><br>Timing to 1 dec. place. Incorrect timing shows as --.-. Take note of the formatting alignment. Average is based on the the number of correct answers. |

To find the number of seconds it takes for user to input the total, make use of the `datetime` library and code segment as follows:

```
import datetime

timeStart = datetime.dateime.now()
total = input('Enter total: ')
timeEnd =datetime.datetime.now()
seconds = (timeEnd – timeStart).total_seconds()
```

The program runs only once. Include a screenshot that shows one program execution.

(15 marks)

(b) Write another program with enhanced features. The program allows a user to play multiple rounds, set the length of digits for each round and the number of tries. A summary is displayed after all the rounds. A sample execution of the program is as follows:

| | |
|---|---|
| How many rounds to play? **2**<br>Enter length of digit for round 1: **3**<br>Enter length of digit for round 2: **4**<br>Number of tries for each round: **3**<br>Round 1<br>Digits: 911<br>Try 1, sum the digits: **11**<br>Digits: 851<br>Try 2, sum the digits: **14**<br>Digits: 482<br>Try 3, sum the digits: **14**<br>Summary<br>Try 1: 911 11 correct 2.2 secs<br>Try 2: 851 14 correct 2.7 secs | Comments<br>No validation required.<br><br>The tries is the same for all rounds.<br>No validation required. |

| | |
|---|---|
| Try 3: 482 14 correct 2.5 secs<br>3/3 correct. Average time is 2.5 secs<br>Hit <Enter> for next round... **<Enter>**<br>Round 2<br>Digits: 9598<br>Try 1, sum the digits: **29**<br>Digits: 2684<br>Try 2, sum the digits:**<Enter>**<br>Try 2, sum the digits: **21**<br>Digits: 1971<br>Try 3, sum the digits: **18**<br>Summary<br>Try 1: 9598 29 Incorrect --.- secs<br>Try 2: 2684 21 Incorrect --.- secs<br>Try 3: 1971 18 correct 4.7 secs<br>1/3 correct. Average time is 4.7 secs<br>Overall summary<br>Round 1: 3/3 correct. Average 2.5 seconds<br>Round 2: 1/3 correct. Average 4.7 seconds | Pause and wait for user to hit <Enter> key for next round.<br><br>If not input, continue to prompt until user enters answer.<br><br>There is no pause after the final round.<br><br>Average time of correct answers only. If all answers are incorrect, Average time should be displayed as --.- secs. |

Include a screenshot showing a program execution of at least 2 rounds with different digit lengths.

This question requires the use of list structure. Dictionary structure should not be used for this question. Make use of functions to make your program modular.

(15 marks)

**Question 4 (25 marks)**

A clothing store maintains inventory on items in their warehouse. All clothing items come in 4 standard sizes, S, M, L, XL. There are 2 files that stores the items and inventory. The first file contains the item code and description and the second file contains the inventories of the various sizes. See appendix A for the contents of the 2 files.

A program is required to track the movement of inventory in and out of the warehouse.

(a) Write a function `loadItems(filename)` with a string filename as parameter. The function reads in the contents from `items.txt`, populates a dictionary and returns the dictionary. The key of the dictionary is the item code and the description is the value. Based on the contents of the file given, the dictionary should look like:

```
items={'C10':'Jogging pants', 'C01':'Airism T shirt', 'C03':'Socks',
'C05': 'Polo T', 'C11':'Shorts'}
```

Ensure all blanks and newline `\n` characters are stripped.

(5 marks)

(b) Write a function `loadInventory(filename)` with a string filename as parameter. The function reads the contents from `inventory.txt`, populates a dictionary and returns the dictionary. Based on the contents of the file given, the dictionary should look like:

```
{'C05':[10,10,5,4], 'C01':[0,20,10,5], 'C11':[10,20,10,1],
'C03':[0,0,10,0]}
```

(5 marks)

(c) Write a function `displayInventory(items, inventory)` where the first parameter is the item dictionary, and the second parameter is the inventory dictionary. Display an inventory report, sorted by item code, in the following format:

```
Inventory Report
C01 Airism T shirt
 S  0
 M  20
 L  10
 XL 5
Total quantity=35

C03 Socks
 S  0
 M  0
 L  10
 XL 0
Total quantity=10
```

(Only a sample of 2 items shown.)

Note: Only partial credit will be given if selection statements are used for this function.

(5 marks)

(d) Write and develop the main function that has a menu as follows:

```
1. Add new item
2. Update inventory
3. Inventory report
4. Exit
Enter option :
```

The main function loads the 2 dictionaries using the functions in part (a) and (b) before the menu is displayed. The description of each of the menu options is as follows:

Option 1 Add new item

This option allows for adding of new items in the warehouse. Prompt for item code. Ensure that item code is not a duplicate of the existing item codes. This is followed by the description and the 4 sizes. A sample input is as follows:

```
Enter item code: C11
Item code already exists!
Enter item code: C88
Enter description: Pullover
Enter the quantities (S, M, L, XL): 12,10,5,0
Inventory added!
Enter item code: <Enter>
```

Note the input requirements:

- Item code may already exist, display an error message and prompt for the code again.

- This option should allow for many items to be added. Use `<Enter>` key as indication of end of input.
- Quantities for the sizes are integers entered with commas in between. No validation of the digits and format is required.

Option 2 – Update Inventory
Input the item code. If the item exists, the description and existing quantity is displayed, followed by the size and quantity. This quantity is added to the stock level for that particular size. A sample program execution is as follows:

```
Enter item code: C29
Item not found!
Enter item code: C11
Item description: Shorts
Current inventory (S, M, L, XL): [10, 20, 10, 1]
Enter size (S, M, L, XL): XL
Enter quantity: 20
C11 size XL quantity updated to 21
```

If item code is not in the dictionary, display 'Item not found'. Prompt for item code again. There is no need to validate input for size and quantity. This option allows for only 1 update. It does not prompt for another item code upon successful update.

Option 3 – Inventory report
Print the complete inventory report as shown in part (c).

(10 marks)

## Appendix A

Assume there are no errors in the contents of the 2 files.

Items.txt

The item code and the description are on separate lines.

```
C10
Jogging pants
C01
Airism T shirt
C03
Socks
C05
Polo T
C11
Shorts
```

Inventory.txt

The item code and the sizes are separated by : followed by the 4 sizes separated by commas.

```
C05:10,10,5,4
C01:0,20,10,5
C11:10,20,10,1
C03:0,0,10,0
```

---- END OF ASSIGNMENT ----