University of South Australia

UniSA STEM

COMP 1039 Problem Solving and Programming

Practical 7

Create a separate file in order to complete each of the following exercises (refer to practical 1 or ask your supervisor if you are having problems doing so). Your files should include appropriate comments consisting of the following – file name, your details, version number and date, brief program description and the University's academic misconduct statement.

Question 1

Write a function called `determine_grade` that accepts a test score as a parameter and returns the corresponding letter grade for the score based on the following grading scale:

Test Score	Grade
0 – 39	F2
40 – 49	F1
50 – 54	P2
55 – 64	P1
65 – 74	C
75 – 84	D
85 - 100	HD

Show how you would call function `determine_grade()`.

Question 2

Design a program that asks the user to enter a series of 20 numbers. The program should store the numbers in a list and then display the following data:

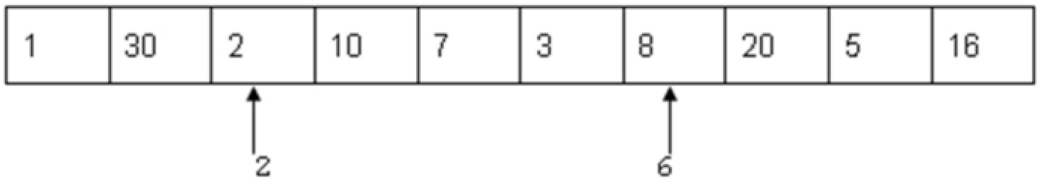
- The lowest number in the list
- The highest number in the list
- The total of the numbers in the list
- The average of the numbers in the list

You **must** use a loop(s) in your solution. You **must not** use the list methods or any built-in methods in order to find the highest, lowest, total and average values.
(Gaddis, Programming Exercises, Chapter 7).

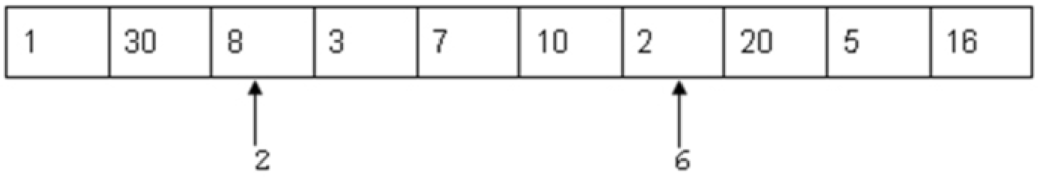
Question 3

Write a program that will prompt for and read ten (10) numbers and store the numbers in a list called `number_list`. The program will then prompt for and read a `top` and `bottom` index value. Your program should then reverse the numbers that are stored in the `number_list` between the top and bottom index (inclusive). You **must** use a loop in your solution. You **must not** use slicing[:], and you must not use list methods.

For example: if `number_list` has the following contents:



and `bottom` is 2 and `top` is 6, then after the reverse has been performed, the contents of `number_list` will be:



Note: That all integers between the `top` and `bottom` index values have also been reversed! Remember that lists are indexed starting at zero.

Question 4

Place your solution for question 3 in a function called `reverse_list`. Function `reverse_list` accepts as parameters, the list of numbers to reverse, a top and bottom index value. The function should then reverse the numbers that are stored in the list (passed in as a parameter) between the top and bottom index values (inclusive). Show how you would call function `reverse_list()`.

Question 5

Write a function called `sing(person)` that sings happy birthday to the person passed in as a parameter. The sing function must call the happy() function defined below.

```
def happy():
    print("Happy Birthday to you!")
```

When the sing(person) function is called like so:

```
sing("Elmer")
```

The function should produce the following output (the Happy Birthday song):

```
Happy Birthday to you!
Happy Birthday to you!
Happy birthday, dear Elmer.
Happy Birthday to you!
```

Question 6 - A game of rock-paper-scissors… revised version:
The following exercise builds on the program written in practical 6, question 1 (if you have not completed the exercise or are having difficulties with it, you may refer to and use the solution available on the course website).

Modify your solution (or the sample solution provided) to the game of rock-paper-scissors so that the solution uses lists.

Please note: The input and output must remain the same, only modify the implementation to make use of lists. For example:

```
selection = ['Rock', 'Paper', 'Scissors']
```

Question 7
Write a function called `generate_password()` that creates and returns a random password consisting of ASCII letters. The function should accept the desired length of the password as an integer parameter.

Question 8
Write code to call the function `generate_password()` (implemented in question 7 above) such that the user can create as many passwords as they like.

Question 9
Write a Python program that will read a file called `'seinfeld.txt'` and display the contents to the screen. Use method `read()` to do this. The `read()` method is used to read the whole file into a string. Hint: Please refer to slide 9 of week 11 reading slides (File Input and Output) for help with this if you are stuck.

Modify this program so that it also writes the contents to a file called `'seinfeldOut1.txt'`. Hint: Please refer to slide 16 of week 11 reading slides (File Input and Output) for help with this if you are stuck.

Sample solution:

```
infile = open("seinfeld.txt", "r")
outfile = open("seinfeldOut1.txt", "w")

string = infile.read()

print(string)
outfile.write(string)

infile.close()
outfile.close()
```

Question 10
Write a Python program that will read a file called `'seinfeld.txt'` and display the contents to the screen. Use method `readlines()` to do this. The `readlines()` method returns a list containing all the lines of the data in the file. Hint: Please refer to slide 11 and 12 of week 11 reading slides (File Input and Output) for help with this if you are stuck.

Modify this program so that it also writes the contents to a file called `'seinfeldOut2.txt'.` Hint: Please refer to slide 16 of week 11 reading slides (File Input and Output) for help with this if you are stuck. `'seinfeldOut2.txt'` should look like `'seinfeld.txt'`.

Question 11
part a)
Write a Python program that will read a file called `'seinfeld.txt'` and display the contents to the screen. Use the file iteration technique which loops (reading one line at a time) until the end of file is encountered as described in slides 13 and 14 of week 11 reading slides (File Input and Output).

Modify this program so that it also writes the contents to a file called `'seinfeldOut3a.txt'.` Hint: Please refer to slide 16 of week 11 reading slides (File Input and Output) for help with this if you are stuck. `'seinfeldOut3a.txt'` should look like `'seinfeld.txt'`.

part b)
Write a Python program that will read a file called `'seinfeld.txt'` and display the contents to the screen. Use the file iteration technique which loops over the file object, that is, going through a file line by line as described in slide 15 of week 11 reading slides (File Input and Output).

Modify this program so that it also writes the contents to a file called `'seinfeldOut3b.txt'.` Hint: Please refer to slide 16 of week 11 reading slides (File Input and Output) for help with this if you are stuck. `'seinfeldOut3b.txt'` should look like `'seinfeld.txt'`.

Question 12 - Let's revisit the game of rock-paper-scissors (yup again)!

The following exercise builds on the program written in practical 6 (game of rock, paper and scissors) (if you have not completed the exercise or are having difficulties with it, you may refer to and use the solution available on the course website).

In a file called `rps_game.py`, place the solution that you will modify (your own solution or the solution obtained from the course website).

1. Modify the game of rock-paper-scissors that you implemented in practical 6 to ensure that your solution defines and uses the functions described below:

    a. Write a function called `generate_number` to generate a random number in the range (1-3). This function accepts no parameters and return a random number in the range (1-3) inclusive.

    b. Write a function called `determine_winner`. This function accepts the computer's choice and the user's choice as parameters. The function determines and displays the winner to the screen. The function does not return anything.

c. Write a function called `play_game` that allows the user to play games of rock, paper and scissors against the computer until he/she does not wish to continue playing (enters 'n' when prompted to '`Play again (y|n)?`'. This function calls functions `display_details()`, `generate_number()`, `get_choice()`, and `determine_winner()`. Note: functions `display_details()` and `get_choice()` were written in practical 6. The `play_game` function should also count how many games the user played and return the total number of games played (by the user) from the function.

2. Call function `play_game` to ensure that it is working correctly. Display the number of games played to the screen.

3. Now that you know the `play_game` function is working correctly, comment out the following code and save the file.

```
# Call function play_game
###num_games = play_game()

###if num_games == 1:
###   print("\n\nYou played", num_games, "game!")
###else:
###   print("\n\nYou played", num_games, "games!")

###print('\nThanks for playing!')
```

4. Create another file called `prac07_ex12.py`. Import the `rps_game` module and call function `play_game()` as seen below. You may wish to read the material on modules posted on the course website under the week 11 or assessment tab.

```
import rps_game

# Call function play_game
num_games = rps_game.play_game()

if num_games == 1:
   print("\n\nYou played", num_games, "game!")
else:
   print("\n\nYou played", num_games, "games!")

print('\nThanks for playing!')
```

Checkpoint: Please make sure your supervisor sees both your program output and code.

Question 13

Describe what the following function does. You may check your answer using Python.

```
def my_function(number):
    new_list = []
    index = 2
    while index < number:
        if number%index == 0:
            new_list.append(index)
        index += 1
    return new_list
```

Part II: More Debugging Practise

Use the debugging techniques described in the debugging slides and video in order to identify and fix the bugs in the following code. In particular, start by using one of the most powerful debugging tools… the humble print statement. Please ensure that you also run the IDLE debugger over the code. The following code (exercises 1, 2, 3 and 4) is available from the course website (this will save you from having to type it in).

Exercise 1

Find and fix the bug in the following code:

```
numbers = [7, 3, 9, 2, 4]
count = 0
sumNo = 0

for num in numbers:
    sumNo = num
    count = count + 0

print("Average of numbers is: ", sumNo / count)
```

The above code should display the follow output to the screen:

```
Average of numbers is:  5.0
```

Exercise 2

Find and fix the bug in the following code:

```
def sum_numbers(no1, no2, no3):
    return no1 + no2 + no3


number1 = 2
number2 = 3
number3 = 4
result = 0

print("Sum of numbers is: ", result)
```

The above code should display the follow output to the screen:

```
Sum of numbers is:  9
```

Exercise 3

Find and fix the bug in the following code:

```
def maximum( no1, no2, no3 ):
    maxValue = no1

    if ( no2 > maxValue ):
        maxValue = no2

    if ( no3 > maxValue ):
        maxValue = no3

    return maxValue


num1 = 4
num2 = 1
num3 = 7

result = maximum(num1, num2, num2)

print('Maximum is: ', result)
```

The above code should display the follow output to the screen:

```
Maximum is:  7
```

Exercise 4

Find and fix the bug in the following code:

```
MAX_VERSES = 100      # maximum number of verses
numberOfVerses = 0    # users choice of number of verses
currentVerseNo = 0    # stores current verse number
answer = 0            # user's response


# Prompt and read whether the user wishes to sing a song.
answer = input("Would you like to sing a song? ")


# While user continues to enter yes.
while answer == 'Y' or answer == 'y':


    # Prompt and read how many verses the user wishes to sing.
    numberOfVerses = int(input("\nHow many verses of the song do you wish to sing?
"))

    # Validate input. Can't be less than or equal to zero or grater than 100.
    while numberOfVerses <= 0 or numberOfVerses > MAX_VERSES:
        print("Not possible my friend ... \n")
        numberOfVerses = int(input("\nHow many verses of the song do you wish to
sing? "))


    # Always start at 100 bottles of beer on the wall.
    currentVerseNo = MAX_VERSES

    # Loop number of verses times.
    for k in range(numberOfVerses):

       # Display verse.
       if currentVerseNo == 1:
         print(currentVerseNo, "bottle of beer on the wall")
         print(currentVerseNo, "bottle of beer")
         print("If one of those bottles should happen to fall")
         print("No bottles of beer on the wall!!\n")
       else:
         print(currentVerseNo, "bottles of beer on the wall")
         print(numberOfVerses, "bottles of beer")
         print("If one of those bottles should happen to fall")

         currentVerseNo = currentVerseNo - 2

         if currentVerseNo == 1:
            print(currentVerseNo, "bottle of beer on the wall\n\n")
         else:
            print(currentVerseNo, "bottles of beer on the wall\n\n")

# Prompt again.
answer = input("\nThat was fun! Would you like to sing again? ")
```

The above code should display the follow output to the screen:

```
Would you like to sing a song? y

How many verses of the song do you wish to sing? 2
100 bottles of beer on the wall
100 bottles of beer
If one of those bottles should happen to fall
99 bottles of beer on the wall


99 bottles of beer on the wall
99 bottles of beer
If one of those bottles should happen to fall
98 bottles of beer on the wall



That was fun! Would you like to sing again? y

How many verses of the song do you wish to sing? 3
100 bottles of beer on the wall
100 bottles of beer
If one of those bottles should happen to fall
99 bottles of beer on the wall


99 bottles of beer on the wall
99 bottles of beer
If one of those bottles should happen to fall
98 bottles of beer on the wall


98 bottles of beer on the wall
98 bottles of beer
If one of those bottles should happen to fall
97 bottles of beer on the wall



That was fun! Would you like to sing again? n
```

Optional Practical Work: **Starting with the graphics module…**

The following practical work is optional. The `graphics.py` *and* `game.py` *modules will not be assessed.*

Python has a module called **Tkinter** that provides facilities for developing graphical applications, including graphical user interfaces.

Tkinter is relatively complex to use for beginner programmers, so we will use a simpler graphics package that is built on top of Tkinter.

This module, called graphics, was written by John Zelle for his book "Python Programming: An introduction to Computer Science". The web site for the book is http://mcsp.wartburg.edu/zelle/python.

Download a copy of Python 3.x version of the module `graphics.py`, which you should include in the same directory as you use for the exercises in this practical. A short document is also available for download. You should download this document and have it available in a window when doing this practical.

Creating a window

You need to import the module graphics to use the facilities of the module. The GraphWin object which is used to create a graphics window in which you can draw.
When you create a GraphWin object, you need to provide:

- A title for the window – a string value
- The width of the window in pixels – a positive integer value
- The height of the window in pixels – a positive integer value

Example:

```
import graphics

win = graphics.GraphWin("Drawing Window", 300, 300)
```

Exercise 1
Write a python program called prac701.py that creates a window that is 500 pixels wide and high and has a title "Exercise 1".

Your solution will most likely create the window, then finish, leaving the window on display. You may need to restart the Python shell (there is a menu item to do this in Idle) to clear this window.

To stop the program from finishing, you can wait for the user to perform some action, such as clicking inside the window. If the displayed window is referred to by a variable called win (as in the example above), you can wait for a mouse click and then finish using:

```
win.getMouse()
win.close()
```

Scroll to the end of this practical for a sample solution if you are having difficulties with this.

The GraphWin object has a number of methods, including:

setBackground(color)
    Sets the background colour.
close()
    Close the on-screen window - any further operations will raise a GraphicsError exception.
getMouse()
    Waits for user to click in the window and returns the mouse coordinates as a Point object.

Note that unless you modify the coordinate system using the method setCoords(), the top left hand corner has coordinates (0, 0), the bottom right hand corner has coordinates (300, 300) – given the window is 300 pixels wide and 300 pixels high.

Exercise 2

Try the following:

- Set the background colour to white
- Set the background colour to black

Scroll to the end of this practical for a sample solution if you are having difficulties with this.

Graphics Objects

There are graphics objects for Point, Line, Circle, Oval, Rectangle, Polygon and Text.
All of these objects support the following methods:

- setFill(color)
- setOutline(color)
- setWidth(pixels) (does not work for Point)
- draw(graphicsWindow)
- undraw()
- move(dx,dy)
- clone()

Point(x,y)
Construct a point at given coordinates.
Can use getX() and getY() to return the coordinate values.

Line(point1, point2)
Constructs a line between the two points.
Method setArrow(string) to change line ends (with string having possible values 'first', 'last', 'both', 'none'.
The default is 'none'.
Have methods getCenter() and getP1(), getP2().

Circle(centerPoint, radius)
Constructs a circle with given center point and radius.
Can get center and radius using getCenter() and getRadius().
Can get bounding box corners using getP1() and getP2().

Rectangle(point1, point2)
Constructs a rectangle having opposite corners at point1 and point2.
Methods getCenter(), getP1(), getP2().

Oval(point1, point2)
Constructs an oval in the bounding box determined by point1 and point2.
Methods getCenter(), getP1(), getP2().

Polygon(point1, point2, ...)
Constructs a polygon through points.
getPoints() returns a list of the points.

Text(anchorPoint, string)
Constructs a text object that displays the given string centered at anchorpoint. The text is displayed horizontally.

See the document graphics.pdf for a full description of the graphics objects.

You can construct a Point object with coordinates (x, y) using Point(x, y).

For example:

The following code creates a Point object at coordinates (250, 250).
```
point = graphics.Point(250,250)
```

The following code changes the colour from the default (which is black) to red.
```
point.setFill("red")
```

The following code draws the point object to the graphics window.
```
point.draw(win)
```

Exercise 3
Set the background to black.
Draw a yellow point at coordinate 250, 250.

Scroll to the end of this practical for a sample solution if you are having difficulties with this.

Exercise 4
Use the method randint() to draw yellow points at 100 random positions inside the window (generate random integer values between 0 and 500 for both the x and y positions.

Scroll to the end of this practical for a sample solution if you are having difficulties with this.

Exercise 5
Modify the point drawing to draw points of random colours.
Refer to section 7 of the graphics.pdf document for a discussion on generating colours.

Scroll to the end of this practical for a sample solution if you are having difficulties with this.

Exercise 6
Place the code that generates a random colour in a function called `gen_colour`. Modify your code so that it now calls the function `gen_colour` in your code in order to generate a random colour.

Scroll to the end of this practical for a sample solution if you are having difficulties with this.

Exercise 7
In a new Python file, write a program that creates a window, sets the title to 'Exercise 7', sets the background to white and draws a red Circle at position (250, 250) with radius of 20 pixels.

Scroll to the end of this practical for a sample solution if you are having difficulties with this.

Exercise 8
Modify the program so that it now generates 1000 circles at random positions and of random colours inside the window.

Scroll to the end of this practical for a sample solution if you are having difficulties with this.

Exercise 9
You may even like to modify your code further by creating circles of random sizes.

Exercise 10
Experiment with creating other shapes and drawing them to the window as you have with Point and Circle objects above.

Optional Practical Work (graphics module) Sample Solutions

Exercise 1

```
import graphics

win = graphics.GraphWin("Exercise 1", 500, 500)
win.getMouse()
win.close()
```

Exercise 2

```
import graphics

win = graphics.GraphWin("Exercise 1", 500, 500)

win.setBackground("white")
# or… win.setBackground("black")

win.getMouse()
win.close()
```

Exercise 3

```
import graphics

win = graphics.GraphWin("Exercise 1", 500, 500)

win.setBackground("black")

point = graphics.Point(250,250)
point.setFill("yellow")
point.draw(win)

win.getMouse()
win.close()
```

Exercise 4

```
import graphics
import random

win = graphics.GraphWin("Exercise 1", 500, 500)

win.setBackground("black")

for i in range(100):
    x = random.randint(0, 500)
    y = random.randint(0, 500)
    point = graphics.Point(x, y)
    point.setFill("yellow")
    point.draw(win)

win.getMouse()
win.close()
```

Exercise 5

```
import graphics
import random

win = graphics.GraphWin("Exercise 1", 500, 500)

win.setBackground("black")

for i in range(100):
    x = random.randint(0, 500)
    y = random.randint(0, 500)
    point = graphics.Point(x, y)
    colour = graphics.color_rgb(random.randint(0, 255),
                                random.randint(0, 255),
                                random.randint(0, 255))
    point.setFill(colour)
    point.draw(win)

win.getMouse()
win.close()
```

Exercise 6

```
import graphics
import random

def gen_colour():
    colour = graphics.color_rgb(random.randint(0, 255),
                                random.randint(0, 255),
                                random.randint(0, 255))
    return colour


win = graphics.GraphWin("Exercise 1", 500, 500)

win.setBackground("black")

for i in range(100):
    x = random.randint(0, 500)
    y = random.randint(0, 500)
    point = graphics.Point(x, y)
    point.setFill(gen_colour())
    point.draw(win)

win.getMouse()
win.close()
```

Exercise 7

```
import graphics

win = graphics.GraphWin("Exercise 7", 500, 500)

win.setBackground("white")

point = graphics.Point(250, 250)
circle = graphics.Circle(point, 20)
circle.setFill('red')
circle.draw(win)

win.getMouse()
win.close()
```

Exercise 8

```
import graphics
import random

def gen_colour():
    colour = graphics.color_rgb(random.randint(0, 255),
                                random.randint(0, 255),
                                random.randint(0, 255))
    return colour


win = graphics.GraphWin("Exercise 7", 500, 500)

win.setBackground("white")

for i in range(1000):
    x = random.randint(0, 500)
    y = random.randint(0, 500)
    point = graphics.Point(x, y)
    circle = graphics.Circle(point, 20)
    circle.setFill(gen_colour())
    circle.draw(win)

win.getMouse()
win.close()
```

Please make sure you save and keep all of your practical and assignment work. Please ask your supervisor if you are having difficulties doing so.

End of practical 7.