Name:

Student ID: Cohort:

10.009 The Digital World

Term 3. 2018.

Mid-Term Exam

March 14, 2018 (Wednesday)

Overall Timing: 2:30-5:00pm. Actual Duration: 2:45pm- 4:45pm (2 hours)

Room: Cohort Classroom

Most recent update: March 7, 2018

- Write your name and student ID at the top of this page.
- This exam has two parts. You can secure a maximum of 100 points.
- For Part A (question 1), write your answers **on this exam paper**.
- For Part A (question 2), put your answers in **eDimension**.
- For Part B (questions 3-7), submit your solutions to **Vocareum**. Template files may be provided in Vocareum.
- You may consult all material on your laptop and in your notes. You may also use any book(s) as a reference.
- **You are not allowed to use any Internet accessing or communicating device during the exam.**
- **You are not allowed to consult anyone inside or outside of the classroom other than the proctors in the examination room.**
- Use your desktop IDE to test your programs. Once you are satisfied, enter your program into Vocareum and either save it or submit. In case you decide to save, then you MUST submit it before the end of the exam.
- All answers will be graded manually. You may be able to earn partial credit for questions.
- Good luck!

Summary:

Category	Description	Number of Problems	Total Points
Part A	Written questions	2 (Questions 1-2)	20
Part B	Programming questions	6 (Questions 3-7)	80

Q1	
Q2	
SubTotal	

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN
HONOUR CODE

"As a member of the SUTD community, I pledge to always uphold honourable conduct. I will be accountable for my words and actions, and be respectful to those around me."

Introduction to the SUTD Honour Code
What is the SUTD Honour Code?
The SUTD Honour Code was established in conjunction with the school's values and beliefs, in good faith that students are able to discern right from wrong, and to uphold honourable conduct. It is an agreement of trust between the students and the staff and faculty of SUTD, and serves as a moral compass for students to align themselves to. Being in a university that aspires to nurture the leaders of tomorrow, it calls for students to behave honourably, not just solely in their academic endeavours, but also in everyday life.

What the Honour Code encompasses
Integrity & Accountability
To be honourable is to do what is right even when nobody is watching, and to be accountable for the things one does. One should always be accountable for one's words and actions, ensuring that they do not cause harm to others. Putting oneself in a favourable position at the expense of others is a compromise of integrity. We seek to create a community whereby we succeed together, and not at the expense of one another.

Respect
Part of being honourable is also respecting the beliefs and feelings of others, and to never demean or insult them. Should conflicts arise, the aim should always be to settle them in a manner that is non-confrontational, and try to reach a compromise. We will meet people of differing beliefs, backgrounds, opinions, and working styles. Understand that nobody is perfect, learn to accept others for who they are, and learn to appreciate diversity.

Community Responsibility
In addition to that, being honourable also involves showing care and concern for the community. Every individual has a duty to uphold honourable conduct, and to ensure that others in the community do likewise. The actions of others that display immoral or unethical conduct should not be condoned nor ignored. We should encourage each other to behave honourably, so as to build a community where we can trust one another to do what is right.

Student's signature

[this page is left blank]

Part A

Q.1 [10 points]

(a) After the following code is executed, at line 6, what is seen on the screen is `1.23 aces`.

By explaining what each line of the code below (for lines 1 to 5) does, show how the code below switches the objects assigned to variables x and y.

Your explanation must state when objects are created and their data types, and also how names are assigned to these objects. Diagrams could be helpful in your explanation.

(6 points)

```
1   x = 'aces'
2   y = 1.23
3   z = x
4   x = y
5   y = z
6   print(x, y)
```

(b) When the following code is executed, after line 12, what is seen on the screen is `True`.

i) Using how names are assigned to objects in memory, explain why. Diagrams could be helpful in your explanation.

(3 points)

ii) The intention of the programmer is to create two lists showing the words for 'seven', 'eight' and 'nine' for both Greek and French. State one modification to the code at line 8 so that line 13 prints out the correct output.

(1 point)

```
7   french = [ 'sept', 'huit', 'neuf' ] #the words mean 'seven','eight','nine'.
8   greek = french
9   greek[0] = 'epta' # 'seven'
10  greek[1] = 'okto' # 'eight'
11  greek[2] = 'enea' # 'nine'
12  print(greek is french)
13  print(french, greek)
```

Your Answer:

Your Answer (Continued):

Q.2 [10 points]

This question has several parts. **Enter your answer in eDimension**.

a) Is there anything wrong with the following program? If yes, what is wrong? (2 points)

```
import = int(input("Enter a number: "))
if import==2:
    print("Yes")
else:
    print("No")
```

b) If `break` is removed from the following program and the program is run, what will be printed out? (2 points)

```
my_string = "Computing"
for character in my_string:
     print(character)
     if character == "u":
          print("Found 'u' :)")
          break
```

c) Look at the following function:

```
def my_function(n):
    return_value = None
    if n == 0 or n == 1:
        return_value = False
    i=2
    while i < n**0.5:
        if n%i==0:
            return_value = False
            break
        i+=1
        return_value = True
    return return_value
```

Let the function be tested with the following statement:

```
print(my_function(37))
```

a. What will be the output? (1 point)
b. Identify the lines of the program which will be executed when the input is 37. Do this by entering the codes from those lines to eDimension. (2 points)

d) In the context of the 1D projects you have completed so far in this course, look at the following function:

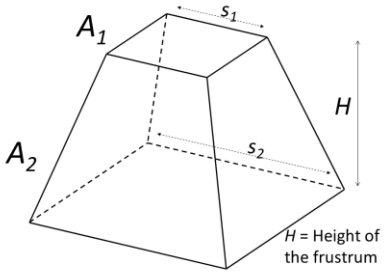
```
def forward(speed, duration):
    robot.wheels(speed, speed)
    robot.sleep(duration)
    robot.wheels(0,0)
```

a. Why is it not necessary to have a `return` statement in this function? (1 point)
b. If we change `robot.wheels(speed, speed)` to `robot.wheels(speed1, speed2)` and the function header is also modified to take in `speed1` and `speed2`, how will the movement of the robot change? You can assume that `speed1` and `speed2` are different and both `speed1` and `speed2` are positive numbers. (1 point)

Part B

Q.3 [10 points]

A frustum is a parallel truncation of a right pyramid. A piece of metal (shown in the Figure below) is in the shape of a frustum with a square base. The side length of the top square is s_1 and the side length of the bottom square is s_2. The height of the frustum is H.



The Volume of the frustum is given by the formula:

$$Volume = \frac{H}{3}\left(A1 + A2 + \sqrt{A1 \times A2}\right)$$

where A1 is the area of the upper square, A2 is the area of the lower square, and H is the height of the frustum.

a) Write a python function `area_square(s)` that takes the side of a square as an input argument s, and returns the area of the square.
b) Write a python function `vol_frustum(top_area, bottom_area, height)` that takes three arguments, a top area, a bottom area and a height in that order, and returns the volume of the frustum.
c) Write a python function `get_volume(s1, s2, height)` that takes three arguments, a top side length, a bottom side length and a height and returns the volume of a frustum based on those dimensions. This function should first call area_square to obtain the two needed areas, and then call vol_frustum to evaluate the volume.

All input arguments and return values are floats. Please round only your **final** output of `get_volume` to three decimal places. Please use `math.sqrt()` to compute the square root in your python code. Note that you get only full marks if your `get_volume` function makes use of the other two functions.

Test Code:	**Output:**

`print('{:.3f}'.format(area_square(2)))`
`print('{:.3f}'.format(area_square(3)))`
`print('{:.3f}'.format(vol_frustum(1,4,2)))`
`print('{:.3f}'.format(vol_frustum(2,2,2)))`
	`4.000`
	`9.000`
	`4.667`
	`4.000`

`print('{:.3f}'.format(get_volume(1,2,2)))`	`4.667`
`print('{:.3f}'.format(get_volume(1.5,3.3,5.0)))`	`30.150`
`print('{:.3f}'.format(get_volume(3.6,6.4,4.0)))`	`102.613`

Q.4 [10 points]

A **square matrix** is a matrix of size *n* x *n*, i.e. *n* rows and *n* columns.

Given a square matrix **M** of size $1 \leqslant n \leqslant 3$, the **determinant** of **M**, denoted **det(M)**, is calculated using the following algorithm:

(1) If *n* = 1, and

$$M = [a]$$

Then **det(M)** = *a*.

(2) If *n* = 2, and

$$M = \begin{bmatrix} \mathrm{a} & \mathrm{b} \\ \mathrm{c} & \mathrm{d} \end{bmatrix}$$

Then **det(M)** = *ad* – *bc*.

(3) If *n* = *3*, and

$$M = \begin{bmatrix} \mathrm{a} & \mathrm{b} & \mathrm{c} \\ \mathrm{d} & \mathrm{e} & \mathrm{f} \\ \mathrm{g} & \mathrm{h} & \mathrm{i} \end{bmatrix}$$

Then:

$$\det(M) = a \cdot \det\left(\begin{bmatrix} \mathrm{e} & \mathrm{f} \\ \mathrm{h} & \mathrm{i} \end{bmatrix}\right) - b \cdot \det\left(\begin{bmatrix} \mathrm{d} & \mathrm{f} \\ \mathrm{g} & \mathrm{i} \end{bmatrix}\right) + c \cdot \det\left(\begin{bmatrix} \mathrm{d} & \mathrm{e} \\ \mathrm{g} & \mathrm{h} \end{bmatrix}\right)$$

Task: Implement a function `determinant(matrix)` that takes a `matrix` as input (represented as a **nested list**) and returns its **determinant** as output. The function should satisfy the following requirements:

- If the input `matrix` is not of dimension *n* x *n* (for $1 \leqslant n \leqslant 3$), the function should return **`None`**;

- The function is to be implemented without importing any libraries (e.g. numpy), otherwise no marks will be awarded.

Hint: in the n = 3 case, it may be helpful to assign the elements of the matrix to variables, and then use those variables in computing the determinant.

Test code:	**Output:**
`print(determinant([[100]]))`	`100`
`print(determinant([[-5, -4], [-2, -3]]))`	`7`
`print(determinant([[2, -3, 1],` `[2, 0, -1], [1, 4, 5]]))`	`49`
`print(determinant([[0, 3, 5],` `[5, 5, 2], [3, 4, 3]]))`	`-2`
`print(determinant([[23], [-4, 4]]))`	`None`

Q.5 [15 points]

The Newton-Raphson (NR) method is an iterative method that approximates the root of a function. The accuracy of the answer is enhanced in successive iterations.

$$i^{th}\ iteration\text{: } x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}$$

In this problem we are applying NR to solve for x the *n*th root of *num*, i.e. $x = \sqrt[n]{num}$.

For instance, to solve the square root of *num*:

$x^2 = num$
$f(x) = x^2 - num$
$f'(x) = 2x$

You need two create two functions: `nroot` and `nroot_complex`. The function `nroot(n, i, num)` is to determine the root of non-negative `num`. The function `nroot_complex(n, i, num)` to determine the root of negative `num`. The function `nroot_complex` should call `nroot` to do the NR approximation. Note the output should give a constant*1j where j is the imaginary $\sqrt{-1}$. For odd *n* the output should give a negative value instead of constant*1j. This means that:

- When `num` is a non-negative number, `nroot_complex` should give the same result as `nroot`.
- When `num` is a negative number and `n` is even, `nroot_complex` should give a complex number with no real part, and its magnitude is the same as the output of `nroot` when `num` is positive.
- When `num` is a negative number and `n` is odd, `nroot_complex` should give a negative real number, and its magnitude is the same as the output of `nroot` when `num` is positive.

Round the output of `nroot` to 3 decimal places. Use x = 1 as your initial value.

Test Code:	**Output:**
`print(nroot(2,5,2))`	`1.414`
`print(nroot_complex(2,5,-4))`	`2j`
`print(nroot_complex(3,5,-8)`	`-2.0`

[this page is left blank]

Q.6 [Total: 30 points]

In this problem you will write a program to find a path through MRT stations from a starting station to an ending station with a maximum of one interchange.

The overall function is called `find_path` and it takes three arguments: (1) a file object to a text file containing the MRT lines and stations, (2) the starting station, and (3) the ending station.

This function should return a list of stations from a starting station to an ending station with a maximum of one interchange. The problem is decomposed by writing several other functions, described in the following parts.

For simplicity, the information given to you in this question is limited to the North South line and the East West line. Also, the branch line to Changi Airport is treated as a separate line. Hence the three lines are labelled in this question as follows: (1) NorthSouthLine (2) EastWestLine (EW) and (3) EastWestLine (CG).

a) `read_stations(f)`: This function takes in a file object and returns a dictionary. The dictionary has the MRT lines as its keys. The value of each key is a list of stations in that MRT line. A sample text input is shown below.

-- start of the shorter text file --

```
=EastWestLine (EW)=

Pasir Ris, Tampines, Simei, Tanah Merah, Bedok, Kembangan, Eunos, Paya Lebar,
Aljuned, Kallang, Lavender, Bugis, City Hall, Raffles Place, Tanjong Pagar,
Outram Park, Tiong Bahru, Redhill, Queenstown, Commonwealth, Buona Vista,
Dover, Clementi, Jurong East, Chinese Garden, Lakeside, Boon Lay, Pioneer,
Joo Koon, Gul Circle, Tuas Crescent, Tuas West Road, Tuas Link

=EastWestLine (CG)=

Tanah Merah, Expo, Changi Airport

=NorthSouthLine=

Jurong East, Bukit Batok, Bukit Gombak, Choa Chu Kang, Yew Tee, Kranji,
Marsiling, Woodlands, Admiralty, Sembawang, Canberra, Yishun, Khatib, Yio Chu
Kang, Ang Mo Kio, Bishan, Braddell, Toa Payoh, Novena, Newton, Orchard,
Somerset, Dhoby Ghaut, City Hall, Raffles Place, Marina Bay, Marina South
Pier
```

-- end of text file –

The output of the function returns a dictionary as follows:

```
{'EastWestLine (EW)': ['Pasir Ris', 'Tampines', 'Simei', 'Tanah Merah',
'Bedok', 'Kembangan', 'Eunos', 'Paya Lebar', 'Aljuned', 'Kallang', 'Lavender',
```

```
'Bugis', 'City Hall', 'Raffles Place', 'Tanjong Pagar', 'Outram Park', 'Tion
Bahru', 'Redhill', 'Queenstown', 'Commonwealth', 'Buona Vista', 'Dover',
'Clementi', 'Jurong East', 'Chinese Garden', 'Lakeside', 'Boon Lay',
'Pioneer', 'Joo Koon', 'Gul Circle', 'Tuas Crescent', 'Tuas West Road', 'Tuas
Link'],

'NorthSouthLine': ['Jurong East', 'Bukit Batok', 'Bukit Gombak', 'Choa Chu
Kang', 'Yew Tee', 'Kranji', 'Marsiling', 'Woodlands', 'Admiralty',
'Sembawang', 'Canberra', 'Yishun', 'Khatib', 'Yio Chu Kang', 'Ang Mo Kio',
'Bishan', 'Braddell', 'Toa Payoh', 'Novena', 'Newton', 'Orchard', 'Somerset',
'Dhoby Ghaut', 'City Hall', 'Raffles Place', 'Marina Bay', 'Marina South
Pier']}
```

Hint: Notice that the station names have no leading or trailing white spaces and you need to use the str.strip() method to ensure this.

Note: You can skip part (a) and do parts (b) or (c) independently.

(7 pts)

(b) `get_stationline(mrt)` : This function takes in a dictionary of MRT lines (i.e. the output of part (a)). The function returns another dictionary which contains all the stations as its keys. The value for each key is a list of the MRT lines for that particular station. Note that if the station is an interchange, the list should contain all the lines calling at that station. For example, if the function takes in the dictionary as shown above, the output should be as follows. Note that Tanah Merah is an interchange and it has two MRT lines.

```
{'Pasir Ris': ['EastWestLine (EW)'], 'Tampines': ['EastWestLine (EW)'],
'Simei': ['EastWestLine (EW)'], 'Tanah Merah': ['EastWestLine (EW)',
'EastWestLine (CG)'], 'Bedok': ['EastWestLine (EW)',

.

.

.

'Somerset': ['NorthSouthLine'], 'Dhoby Ghaut': ['NorthSouthLine'], 'Marina
Bay': ['NorthSouthLine'], 'Marina South Pier': ['NorthSouthLine']}
```

(7 pts)

(c) `get_interchange(stationline)` : This function takes in a dictionary of stations and their lines (i.e. the output of part (b)). The function returns another dictionary which contains all the interchange stations as its keys. The value for each key is a list of the MRT lines for that particular

interchange stations. For example, taking in the output in part (b) as the input, this function should output the following:

```
{'Tanah Merah': ['EastWestLine (EW)', 'EastWestLine (CG)'], 'City Hall':
['EastWestLine (EW)', 'NorthSouthLine'], 'Raffles Place': ['EastWestLine
(EW)', 'NorthSouthLine'], 'Jurong East': ['EastWestLine (EW)',
'NorthSouthLine']}
```

(7 pts)

(d) `find_path(f, start, end)` : This function takes in three arguments: (1) the file object to a text file containing all the MRT lines and stations, (2) the starting station, and (3) the ending station. The function should return a list of stations starting from the starting station to the ending station with a maximum of one interchange. If there are more than one possible paths, the following considerations should be taken into account:

- If there is a path without changing MRT lines, the result should return this path.
- If the path must involve changing MRT lines, it will return the path with a minimum number of stations and containing only one interchange station.
- If no such path can be found as above, it will return None.

Note: This function must make use of the other three previous functions.

For example,

Test code:	**Output:**

`print('Test 1')`
`f=open('mrt_lines_short.txt','r')`
`ans=find_path(f,'Boon Lay', 'Clementi')`
`print(ans)`
`f.close()`
	`Test 1`
	`['Boon Lay', 'Lakeside', 'Chinese Garden', 'Jurong East', 'Clementi']`

`print('Test 2')`
`f=open('mrt_lines_short.txt','r')`
`ans=find_path(f,'Changi Airport', 'Orchard')`
`print(ans)`
`f.close()`
	`Test 2`
	`None`

`print('Test 3')`
`f=open('mrt_lines_short.txt','r')`
`ans=find_path(f,'Boon Lay', 'Bukit Gombak')`
`print(ans)`
`f.close()`
	`Test 3`
	`['Boon Lay', 'Lakeside', 'Chinese Garden', 'Jurong East', 'Bukit Batok', 'Bukit Gombak']`

print('Test 4') f=open('mrt_lines_short.txt','r') ans=find_path(f,'Tanah Merah', 'Orchard') print(ans) f.close()	Test 4 ['Tanah Merah', 'Bedok', 'Kembangan', 'Eunos', 'Paya Lebar', 'Aljuned', 'Kallang', 'Lavender', 'Bugis', 'City Hall', 'Dhoby Ghaut', 'Somerset', 'Orchard']

(9 pts)

Q.7 [15 points]

The currency of the United Kingdom is made up of pounds (£) and pence (p), where £1 is equal in value to 100p.

At present, there are eight coins in general circulation:

1p, 2p, 5p, 10p, 20p, 50p, £1, and £2

Task: write a function `decompose(pence)`, that takes as input some number of **pence** (as an integer), and returns as output an integer expressing **how many different ways** that the amount can be made up by using the available coins.

For example, `decompose(7)` should return **6**, because 7p can be made up from:

- Seven 1p
- One 2p; five 1p
- Two 2p; three 1p
- Three 2p; one 1p
- One 5p; one 2p
- One 5p; two 1p

Note that the function decompose(pence) can be implemented in a number of different ways, including by using *brute force* (i.e. exhaustive search). However, brute force implementations may only score a maximum of 12 points; the full 15 are only available for more elegant/efficient solutions.

Hint #1: remember that every coin (except 1p) can be replaced with a combination of smaller coins.

Hint #2: if you wish to solve this using brute force, you may find the code `range(amount, -1, -coin)` *useful*; `range(400,-1,-200)`*, for example, generates the sequence* 400, 200, 0.

See test cases on the following page.

Test code:	**Output:**
`print (decompose(1))`	`1`
`print (decompose(5))`	`4`
`print (decompose(7))`	`6`
`print (decompose(130))`	`12337`
`print (decompose(200))`	`73682`
`print (decompose(700))`	`40208370`

End of Exam Paper

[this page is left blank]

[this page is left blank]