Name:

Student ID:

10.009 The Digital World

Term 3. January 26-May 1, 2015.

Midterm Exam

March 18, 2015 (Wed) 2:45-4:45pm. Room: Cohort Classroom

Most recent update: March 6, 2015

- Write your name and ID at the top of this page.
- This exam has three parts. First try to complete Part A and Part B and then move to Part C. You can secure a maximum of 100 points.
- For Part A (questions 1-2), write your answers on this exam paper.
- For Part B and C (questions 3-7), submit your solution to Tutor.
- 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 Idle / Canopy to test your programs. Once you are satisfied, enter your program into Tutor 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	Very short programming questions	3 (Questions 3-5)	30
Part C	Challenging programming questions	2 (Questions 6-7)	50

1	
2	
SubTotal	

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Established in collaboration with MIT

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

Part A

Q.1 [10 points]

In the following code, explain the differences between the statements in (i), (ii), (iii), in particular regarding the effects due to aliasing.

```
import copy
a = [1, 2, [3, 4], 5]

b = a                       # (i)
c = a[:]                    # (ii)
d = copy.deepcopy(a)        # (iii)
```

Your Answer:

Q.2 [10 points]

A student wrote the following Python program to instruct the Finch robot to move forward until an obstacle is detected. In particular, when an obstacle is detected, the Finch robot should stop. What is the potential issue with the code? Suggest how you will fix the issue. (Hint: The issue is related to the logic of the code rather than programming syntax)

```
from time import sleep
from finch import Finch

finch = Finch()

finch.wheels(0.2, 0.2)
leftObstacle, rightObstacle = finch.obstacle()
while leftObstacle==False and rightObstacle==False:
    finch.wheels(0.2, 0.2)

finch.halt()
finch.close()
```

Your Answer:

Part B

Q.3 [5 points]

Write a function `comp(x)` that returns the result:

$x^3 + 4x^2 + 6x + 1$

Here `x` is the input argument.

Sample tests:

Input: `comp(2)`

Output: 37

Input: `comp(3)`

Output: 82

Q.4 [10 points]

Write a function `genList(n1,n2)` that takes `n1`, `n2` as inputs and returns a list that includes all the integers that are **multiple of three** between `n1` and `n2` (inclusive of `n1`, `n2`).

Note that `n1,n2` are integers, and `n1 < n2`.

Sample tests:

Input: `genList(2,12)`

Output:

```
[3, 6, 9, 12]
```

Input: `genList(2,20)`

Output:

```
[3, 6, 9, 12, 15, 18]
```

Q.5 [15 points]

Write a function `matAdd(A, B)` to compute the sum of two matrices `A` and `B`. Specifically, given two `mxn` matrices $A = [a_{ij}], B = [b_{ij}]$, `matAdd(A, B)` returns a new matrix $C = A + B$. That is, $C = [c_{ij}]$ is a `mxn` matrix, $c_{ij} = a_{ij} + b_{ij}$.

You can assume `A` and `B` are always `mxn`. Note that you need to return a new matrix `C` that does not have any aliasing with the input matrices `A` and `B`.

The `mxn` matrices `A`, `B` and `C` are represented by Python nested lists. Each matrix is a nested list that has `m` entries, each entry is a list that represents a row of the matrix.

Sample tests:

Input:

```
A = [[1,2,3], [4, 5, 6]]

B = [[10,20,30], [40, 50, 60]]

C = matAdd(A,B)

print 'A:', A, 'B:', B, 'C:', C
```

Output:

```
A: [[1, 2, 3], [4, 5, 6]] B: [[10, 20, 30], [40, 50, 60]]

C: [[11, 22, 33], [44, 55, 66]]
```

Part C

Q.6 [Total: 35 points]

In this question, you will need to write and submit three functions:

`getSchedule(f)`

`findLength(dictSchedule)`

`findConflict(dictSchedule)`

Note that you can work on these functions in any order. In particular, most of you would find that `findLength()` is easier, while `findConflict()` is difficult. However, you need to read part (a) to understand the problem context.

(a) [10 points]

`getSchedule(f)`**:**

This function takes a file object `f` as input (NOT the name of a file, but a file object). The file referred by `f` contains several lines of data (see the example file and test case below). The file consists of multiple schedules. Each schedule begins with a string, on a line by itself denoting the day of a week. After that the following lines give the start and end times of each event in that day. Each event has the starting time (in hour) and the finishing time (in hour), on a line by itself. The end time is guaranteed to be strictly after the start time. The time is in 24-hour format.

Example file:

```
Wednesday
6 8
Thursday
7 11
10 13
```

This example file contains the schedules for Wednesday and Thursday. Wednesday's schedule has one event: 6:00 to 8:00. Thursday's schedule has two events: 7:00 to 11:00, and 10:00 to 13:00.

In general, the file may contain up to seven schedules for each day of a week. Each schedule contains at least one event.

The function `getSchedule(f)` reads data from the file and returns a dictionary. In the returned dictionary, each entry (key-value pair) represents one schedule. In particular, each entry has the day of the week as the key (string data type), and a list containing all the events as the value corresponding to this key. Each event is a tuple of starting time (int data type) and finishing time (int data type).

Test case (Test data file can be downloaded from SUTD eDimension, under Week-8):

```
Monday
9 11
12 14
11 12
Tuesday
12 16
9 12
15 18
19 21
Wednesday
6 8
Thursday
7 11
10 13
```

The dictionary returned by `getSchedule(f)`:

```
{'Tuesday': [(12, 16), (9, 12), (15, 18), (19, 21)], 'Thursday': [(7, 11), (10, 13)], 'Wednesday': [(6, 8)], 'Monday': [(9, 11), (12, 14), (11, 12)]}
```

Note: the dictionary will be unordered. While grading, your returned dictionary will be automatically ordered before comparing it with the Tutor output.

(b) [10 points]

`findLength(dictSchedule)`:

This function takes as input a dictionary `dictSchedule` created by `getSchedule(f)` and returns a dictionary. The function computes the total length (in hour) of the events for each day (see the test case below). In the returned dictionary, each entry has the day of the week as the key (string data type), and the total length of daily events in hour (int data type) as the value corresponding to this key.

Test case:

Input: `{'Tuesday': [(12, 16), (9, 12), (15, 18), (19, 21)], 'Thursday': [(7, 11), (10, 13)], 'Wednesday': [(6, 8)], 'Monday': [(9, 11), (12, 14), (11, 12)]}`

Output: `{'Tuesday': 12, 'Monday': 5, 'Wednesday': 2, 'Thursday': 7}`

(c) [15 points]

`findConflict(dictSchedule):`

This function takes as input a dictionary `dictSchedule` created by `getSchedule(f)` and returns a dictionary. The function checks if there is any conflict for the events of each day (see the test case below). A schedule has conflict if the duration of any of the events overlaps with another one of the same day.

In the returned dictionary, each entry has the day of the week as the key (string data type). The value corresponding to this key is a boolean value that indicates if there is any event conflict in that day, i.e. `True` if there is at least one conflict, `False` if there is no conflict for the daily events.

Test case:

Input: `{'Tuesday': [(12, 16), (9, 12), (15, 18), (19, 21)], 'Thursday': [(7, 11), (10, 13)], 'Wednesday': [(6, 8)], 'Monday': [(9, 11), (12, 14), (11, 12)]}`

Output: `{'Tuesday': True, 'Monday': False, 'Wednesday': False, 'Thursday': True}`

Q.7 [15 points]

A computer screen display consists of millions of picture elements (or pixels), which light up when you draw on the display. Write a Python function to count the number of lit pixels when a circle is drawn.

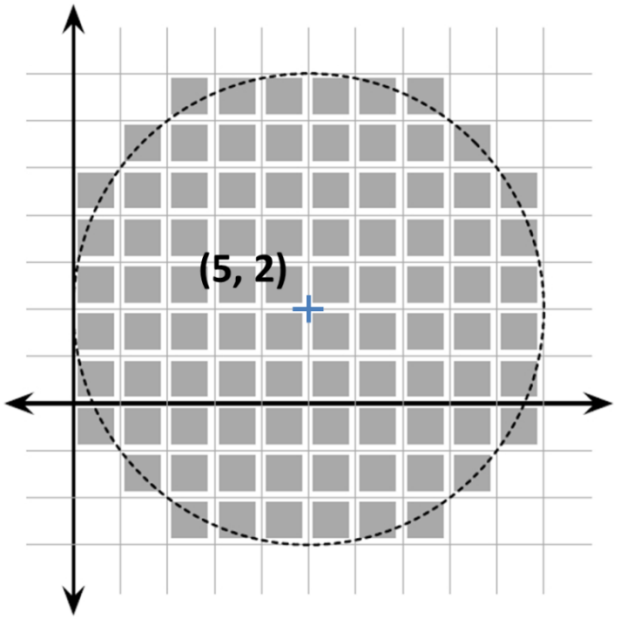
Assume that the display is set on a Cartesian grid and every pixel is a perfect unit square. For example, one pixel occupies the area of a square with corners (0,0), (0,1), (1,1), (1,0).

A circle can be drawn by specifying its center in the grid coordinates and its radius (center coordinates and radius are integers). On the display, a pixel is lit if any part of it is covered by the circle being drawn. However, pixels whose edges or corners are just touched by the circle are not lit.

Write a Python function:

```
countLitPixel(cx, cy, r)
```

Input arguments `cx`, `cy`, `r` are all integers, `r > 0`. Given the center of the circle `(cx,cy)` and its radius `r`, `countLitPixel()` returns the exact number of pixels that are lit when the circle is drawn on the display. For example, a circle with center (5,2) and radius 5 is shown here:



Sample tests:

`countLitPixel(5, 2, 5)` returns 88

`countLitPixel(1,1,1)` returns 4

End of Exam Paper