CS1131 Notes

Computational Thinking I

Revision Paper 2 Worked Solutions

Links of All LQ Notes: LQ Notes Links Document

Topics
Question 1
Question 2
Question 3
Question 4
Question 5

Disclaimer

This is a set of notes going through Revision 2: More practices from NUS High Coursemology, since it seems many of you guys have a few concerns / are struggling a bit with it.

Again, since this contains information from NUS High Coursemology, please **do not** share this set of notes outside of your NUS High Y1 Classmates. Thank you :D

Good luck!

~LQ (25 Sep)

Question 1

Use a for loop to compute the **first 60 terms** of the following series:

$$1 + 2^3 + 3^3 + 4^3 + 5^3 + \ldots$$

Store the final answer in the variable named **resultA** and print out the result.

Note that 0 marks will be given for hardcoded solution.

Solution

First, we notice that this is just the sum of the cubes from 1 to 60 (i.e. $1^3 + 2^3 + \ldots + 60^3$).

Let's first define the variable `resultA`, which should start at 0. We use `resultA = 0`.

```
resultA = 0

```

Now, since the sum starts with 1^3 and ends with 60^3, we should loop over the values 1 to 60. In this case, we use `range(1, 61)` because range ends at one number before what's specified.

```
resultA = 0

for n in range(1, 61):

```

Now, since every term should be cubed, before being added to the `resultA` variable, we use the operation `n ** 3`, then add that to resultA.

```
resultA = 0

for n in range(1, 61):
    resultA = resultA + n ** 3

```

Finally, we print out resultA.

```
resultA = 0
```

```
2
3 for n in range(1, 61):
4     resultA = resultA + n ** 3
5
6 print(resultA)
7
```

And that’s our solution.

Question 2

Use a for loop to compute the first *n* terms of the series shown below.
Note that *n* is an integer input (from user) that is greater or equal to 1.

$$\frac{1}{2}-\frac{2}{3}+\frac{3}{4}-\cdots$$

Store the final answer in the variable named **resultB** and print out the result.

Sample output:

```
Enter n: 3
0.5833333333333334
```

Solution

Let's first define the variable `resultB`, which should start at 0. We use `resultB = 0`.

```
1  resultB = 0
2
```

We then take a user input for n, and store it as a variable. Do recall that the `input()` function gives a string (i.e. text), so we should convert it into an integer before we can do much with it.

```
1  resultB = 0
2  n = int(input("Enter n: "))
3
```

Now, we see each term has alternating sign (i.e. plus, then minus, then plus, and so on…), we should define a variable `sign` to keep track of that. Since the first term is positive, we let `sign` be `1`.

```
1  resultB = 0
2  n = int(input("Enter n: "))
3  sign = 1
4
```

Since the first term is $\frac{1}{2}$ and the last term is $\frac{n}{n+1}$, there are n terms in total, so we use `range(1, n+1)`.

```
resultB = 0
n = int(input("Enter n: "))
sign = 1

for i in range(1, n + 1):

```

Now, we recognise that the i-th term is just $\frac{i}{i+1}$, multiplied by the `sign` variable. Then, we add this to `resultB`.

```
resultB = 0
n = int(input("Enter n: "))
sign = 1

for i in range(1, n + 1):
    resultB = resultB + i * sign / (i + 1)

```

But now recall that the sign alternates after every term. Hence, we need to change it from 1 to -1 or -1 to 1. In other words, we multiply `sign` by -1.

```
resultB = 0
n = int(input("Enter n: "))
sign = 1

for i in range(1, n + 1):
    resultB = resultB + i * sign / (i + 1)
    sign = (-1) * sign

```

Lastly, we print resultB.

```
resultB = 0
n = int(input("Enter n: "))
sign = 1

for i in range(1, n + 1):
    resultB = resultB + i * sign / (i + 1)
    sign = (-1) * sign

print(resultB)


```

Question 3

Use a nested for loop to compute the first *n* terms of the series shown below.
Note that *n* is an integer input (from user) that is greater or equal to 1.

$$1! + 2! + 3! + 4! + \ldots$$

Store the final answer in the variable named **resultC** and print out the result.

Sample output:

```
Enter n: 4
33
```

Solution

Let's first write some code to calculate the factorial of any number. We store the value in a variable called `fact`, which will start at 1.

```
1  fact = 1
2
```

Let's calculate the factorial for a number `i`, but let's not worry about defining `i` first. We can just think of it as a placeholder.

To calculate the factorial of a number i, we must multiply the numbers 1 to i, inclusive. Hence, we use `range(1, n + 1)`.

```
1  fact = 1
2
3  for j in range(1, i + 1):
4
```

For every number `j` in this range, we multiply `fact` by `j` and update the value of `fact`.

```
1  fact = 1
2
3  for j in range(1, i + 1):
4     fact = fact * j
5
```

Now, let's consider how to add all the factorials. We take a user input for n, and define a variable `resultC` as 0.

```
resultC = 0
n = int(input("Enter n: "))

```

Now, we loop through the numbers from 1 to n. Hence, we use `range(1, n + 1)`.

```
resultC = 0
n = int(input("Enter n: "))

for i in range(1, n + 1):

```

We then include the code we wrote just now. See how now we have defined the variable `i`? This shows that in coding, sometimes you can leave things hanging for a while, then go back to it.

```
resultC = 0
n = int(input("Enter n: "))

for i in range(1, n + 1):
    fact = 1

    for j in range(1, i + 1):
        fact = fact * j

```

Then, we add the factorial we calculate to `resultC`.

```
resultC = 0
n = int(input("Enter n: "))

for i in range(1, n + 1):
    fact = 1

    for j in range(1, i + 1):
        fact = fact * j

    resultC = resultC + fact

```

Lastly, we print `resultC`.

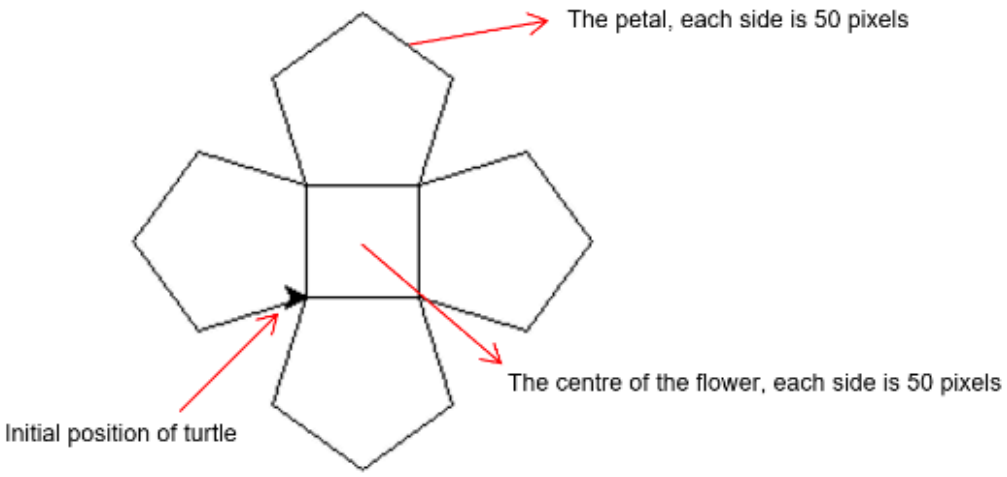
```
resultC = 0
n = int(input("Enter n: "))

for i in range(1, n + 1):
    fact = 1

    for j in range(1, i + 1):
        fact = fact * j

    resultC = resultC + fact

print(resultC)

```

And we are done :D

Question 4

Please DO NOT import turtle for this question.

Consider the diagram shown below:



We name the pattern above a flower.

The flower is made up of a centre piece, in this case, a square. Each side of the square is 50 pixels in length.

The flower has petals, in this case, 4 pentagons. Each side of the pentagon is 50 pixels in length.

Write relevant python code to draw the diagram shown above.

Solution

First, let's import turtle. (Read instructions carefully, since while coding you should still include this line, but don't copy it into the submitted copy)

```
1 import turtle
2
```

Now, to draw a pentagon, let's use the code to draw any n-sided polygon.

Note that the value 50 is given, and the angle 72° is obtained by 360° ÷ 5 = 72°.

```
1 import turtle
2
3 for n in range(5):
```

```
    turtle.forward(50)
    turtle.right(72)

```

Now, we note that we have to draw 4 pentagons, and between each pentagon, we need to move the turtle forward by 50, then turn it left by 90 degrees.

```
import turtle

for m in range(4):
    for n in range(5):
        turtle.forward(50)
        turtle.right(72)
    turtle.forward(50)
    turtle.left(90)

```

Again, remember to delete `import turtle`, then copy to coursemology if the question requires you to.

We should also check that the drawing looks right, and after running the code, we see that it is indeed correct. So, we're done :D

Question 5

Please DO NOT import turtle and random for this question.

We would now like to make relevant changes to the code in Question 4 such that the drawing can be made more generic.

Your program will first ask user to input the following:

- Number of petals in the flower
- Length of each side of the petal

Next, generate a random integer number between 3 to 10 (inclusive of both) and store in a variable **petalSides**. This variable represents the number of sides each petal should have.

Finally, draw the required graphics based on the inputs stated above.

Solution

First, let's import turtle and random. (Read instructions carefully, since while coding you should still include these lines, but don't copy it into the submitted copy)

```
1  import turtle
2  import random
3
```

Now, let's generate the random variable `petalSides`. Recall `random.randrange` includes the lower limit, but does not include the upper limit. Hence, to generate a random number 3 to 10 inclusive, we write `random.randrange(3, 11)`.

```
1  import turtle
2  import random
3
4  petalSides = random.randrange(3, 11)
5
```

We then get two user inputs, `petalNum` and `length` from the user.

```
1  import turtle
2  import random
3
4  petalSides = random.randrange(3, 11)
5  petalNum = int(input('Enter number of petals: '))
6  length = int(input('Enter length of each side of petal: '))
7
```

Let's now *ahem* "borrow" code from the previous question that we wrote and copy it below, getting rid of the `import turtle`.

```
import turtle
import random

petalSides = random.randrange(3, 11)
petalNum = int(input('Enter number of petals: '))
length = int(input('Enter length of each side of petal: '))

for m in range(4):
    for n in range(5):
        turtle.forward(50)
        turtle.right(72)
    turtle.forward(50)
    turtle.left(90)

```

Now, we should replace some numbers with the variables we have defined. Instead of `m in range(4)`, it should be `m in range(petalNum)`.

Instead of `turtle.forward(50)`, both should be changed to `turtle.forward(length)`

Instead of `n in range(5)`, it should be `n in range(petalSides)`

Instead of `turtle.right(72)`, it should be `turtle.right(360 / petalSides)`

Lastly, instead of `turtle.left(90)`, it should be `turtle.left(360 / petalNum)`

And... we're done with generalisation.

```
import turtle
import random

petalSides = random.randint(3, 10)
petalNum = int(input('Enter number of petals: '))
length = int(input('Enter length of each side of petal: '))

for m in range(petalNum):
    turtle.forward(length)
    for n in range(petalSides):
        turtle.right(360 / petalSides)
        turtle.forward(length)
    turtle.left(360 / petalNum)

```

Again, remember to delete the import statements if the question says so.
Woo! We've finished going through Revision 2. Again, good luck for your CS Exams <3