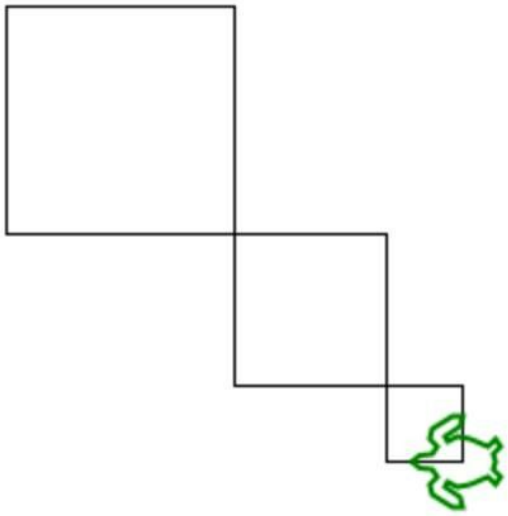
cs101: Computer Programming and Utilization Jan-Apr 2016

Model Answers for the Questions in the Ungraded Lab of the Week 1

For all questions, the turtle orientations need **not** be that given in the figures.

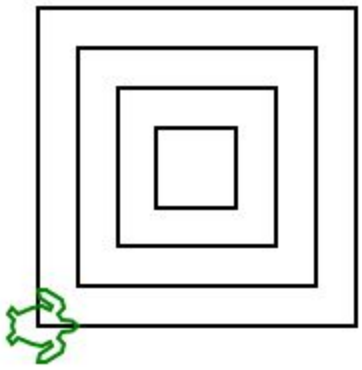
Q1. Write a program to draw the figure shown. The area of the three squares are in the ratio 9:4:1.



```
#include <simplecpp>
main_program{
        int len=3;
        turtleSim();
        repeat(6)
        {
                forward (30*len);
                right(90);
        }
        right(180);
        repeat(6)
        {
                forward (20*len);
                right(90);
        }
        right(180);
        repeat(6)
        {
                forward (10*len);
                right(90);
```

```
        }
        wait(10);
}
```

Q2: Write a program which draws 4 concentric squares. The innermost square should have a side of 40 and each two consecutive concentric squares should be separated by a distance of 20 (the second innermost one will have a side of 80). The figure should look like the following:



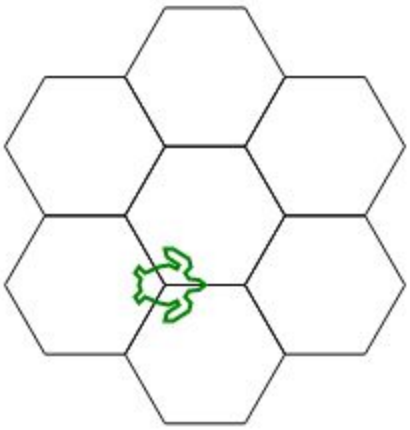
```
#include <simplecpp>
main_program{
    turtleSim();
    repeat(4){
        forward(40);
        left(90);
    }

    penUp();
    right(135);
    forward(20*sqrt(2));
    left(135);
    penDown();

    repeat(4){
        forward(80);
        left(90);
    }

    penUp();
    right(135);
```

```
    forward(20*sqrt(2));
    left(135);
    penDown();

    repeat(4){
        forward(120);
        left(90);
    }

    penUp();
    right(135);
    forward(20*sqrt(2));
    left(135);
    penDown();

    repeat(4){
        forward(160);
        left(90);
    }
}
```

Q3: Draw an N-sided regular polygon surrounded by **N** N-sided regular polygons of same side length such that the polygon in the centre shares a side with the surrounding polygons. Your program should take the number of sides (N) of the regular polygon as input. Consider the diagrams below as examples.

[Limit N <= 8]

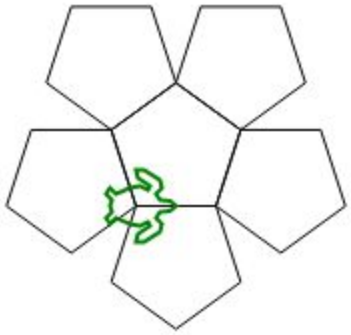
N = 6



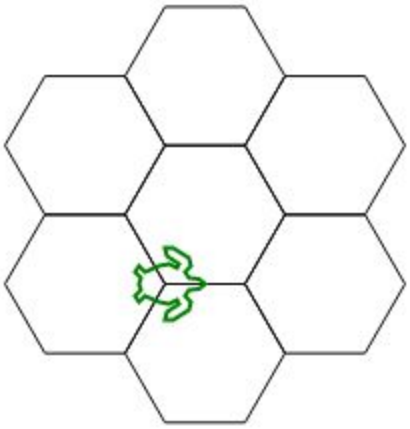
Experimentally fix the side length such that the entire figure should fit within the canvas.

```
#include<simplecpp>
main_program{
        turtleSim();
        int n;
        cin>>n;
        repeat(n){
                repeat(n){
                        forward(40);
                        right(360.0/n);
                }
                forward(40);
                left(360.0/n);
        }
}
```

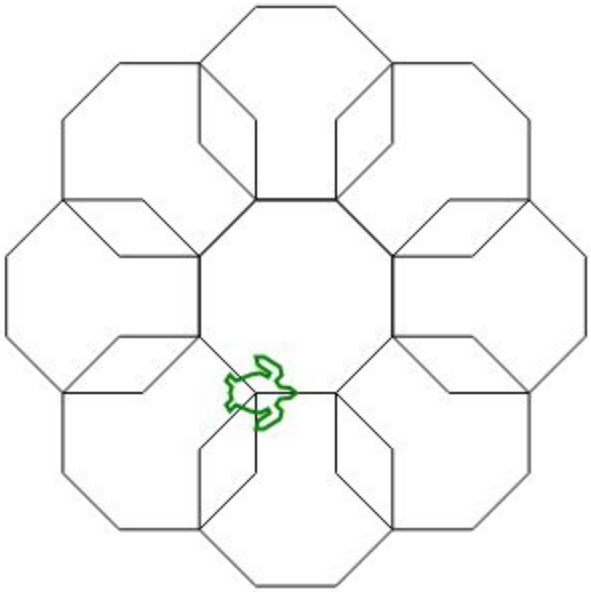
Output in Prutor:
For N = 5



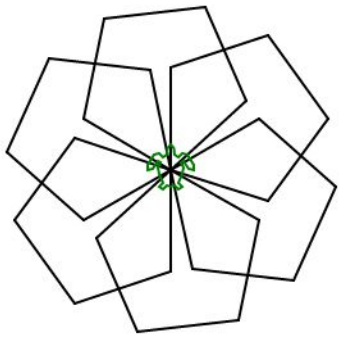
For N = 6



For N = 8



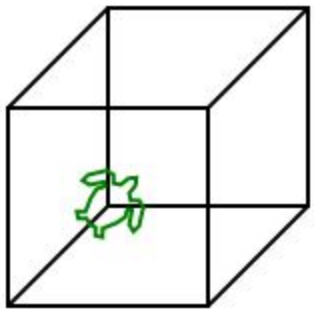
Q4: Draw the following figure consisting of 6 pentagons rotated at 60 degrees with respect to their neighbours.



```
#include <simplecpp>
main_program {
	turtleSim();
	left(90);
	int i = 80;
	int angle = 72;
	repeat(6) {
		forward(i); right(angle);
```

```
                forward(i); right(angle);
                forward(i); right(angle);
                forward(i); right(angle);
                forward(i); right(angle);
                right(360/6);
        }
        wait(200);
}
```

Q5: Draw a cuboid similar to the following figure.

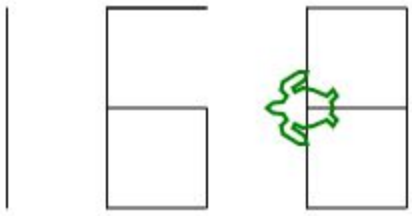


Make sure there are no breaks in the figure. For this you may have to use some of the commands/functions like **sqrt** (square root) described in the class. There is no fixed answer for this question. Fix some angle for the slant lines and use side lengths not exceeding 100.

```
#include <simplecpp>
main_program{
        turtleSim();
        int sideLength = 100;
        repeat(4){
                forward(sideLength);
                right(90);
        }
        penUp();
        forward(sideLength/2);
        left(90);
        forward(sideLength/2);
```

```
        right(90);
        penDown();
        repeat(4){
                forward(sideLength);
                right(90);
        }
        right(135);
        forward(sideLength / sqrt(2));
        left(135);
        forward(sideLength);
        left(45);
        forward(sideLength / sqrt(2));
        right(135);
        forward(sideLength);
        right(45);
        forward(sideLength / sqrt(2));
        right(45);
        forward(sideLength);
        right(135);
        forward(sideLength / sqrt(2));
        wait(30);
}
```

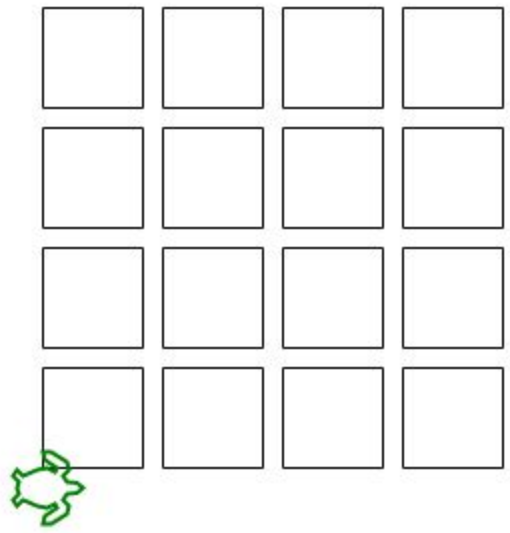
Q6: Draw the last three digits digits of your roll.no using the basic 7 segment display.
Draw all digits using only straight lines as you see on a calculator. Figure below shows the output for number 168



```
#include <simplecpp>
main_program{
        turtleSim();

        left(90);
        forward(100);
        right(90);
        penUp();
        forward(50);
```

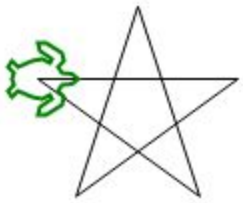
```
        penDown();
        forward(50);
        forward(-50);
        right(90);
        forward(100);
        left(90);
        forward(50);
        left(90);
        forward(50);
        left(90);
        forward(50);
        penUp();
        left(90);
        forward(50);
        left(90);
        forward(150);
        penDown();

        left(90);
        forward(100);
        left(90);
        forward(50);
        left(90);
        forward(100);
        left(90);
        forward(50);
        left(90);
        penUp();
        forward(50);
        left(90);
        penDown();
        forward(50);

        wait(20);
}
```

Q7: Draw a board as follows



```
#include <simplecpp>
main_program{
        turtleSim();
        repeat(4){
            repeat(4){
                repeat(4){
                    forward(50);
                    right(90);
                }
                penUp();
                forward(60);
                penDown();
            }
            penUp();
            right(90);
            forward(60);
            right(90);
            forward(240);
            right(180);
            penDown();
        }
}
```

Q8: Draw a 5 point star like the one given below. What strategy will be required to generalize it to an N-point star? (You do not have to write a program for N-point star).



```
#include <simplecpp>
main_program{
	turtleSim();
	repeat(5){
		forward(100);
		right(144);
	}
	wait(10);
}
```