2a)

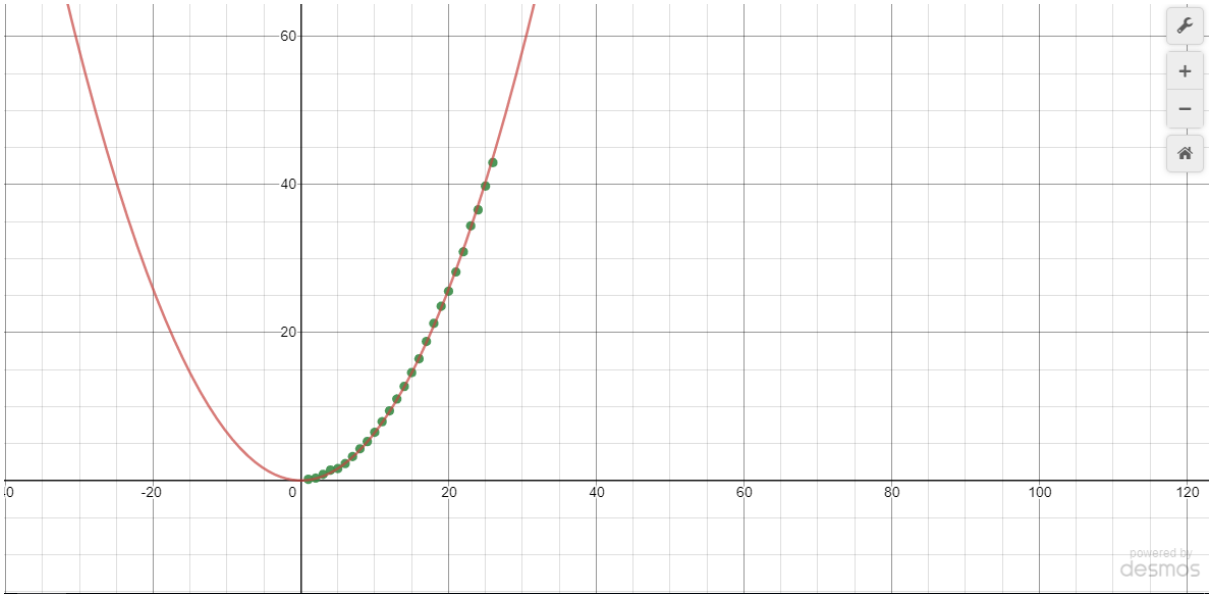
```
public class StringRepeater {
   public String repeatString(String s, int n){
      String result = "";
      for(int i=0; i<n; i++) {
         result = result + s;
      }
      return result;
   }
}
```

Code	**Elementary operations**
String result = "";	1
Int i=0	1
I<n	n+1
I++	n
result = result + s;	Xn
return result	1

T(n) = 1+1+n+1+n+x+1

T(n) = Xn+2n+4

b)



Graph drawer: https://www.desmos.com/calculator

T(N/1000) = Mean/ 10_000_000

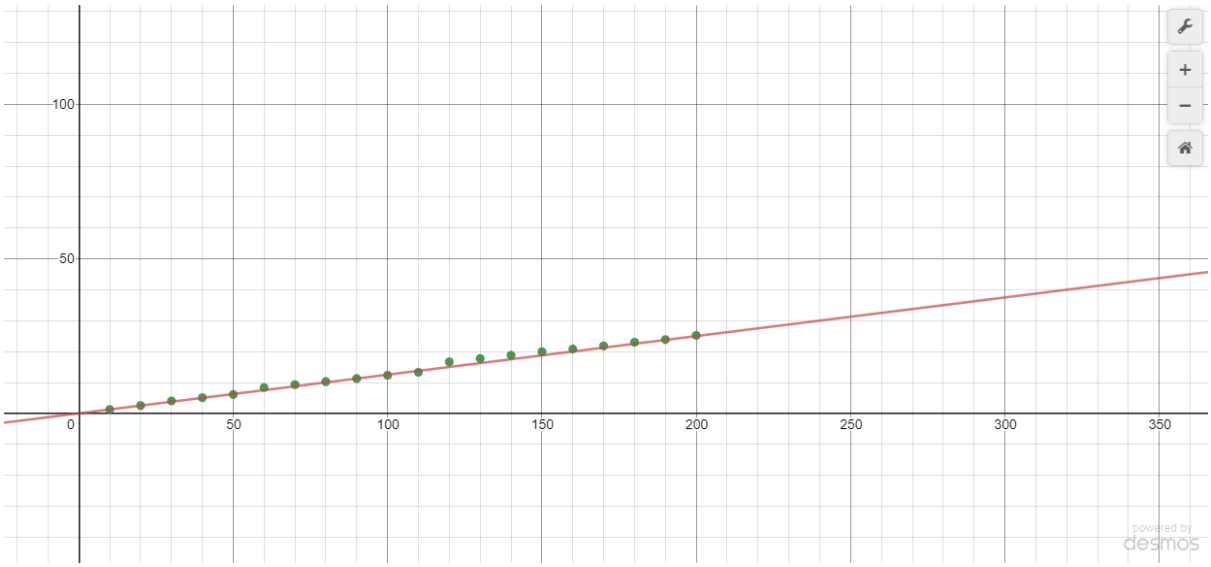
Divided N and the Mean in order to get a sensible looking graph.

T(1) = 0.15807381599999998	T(14) = 12.713327624
T(2) = 0.15807381599999998	T(15) = 14.579112501
T(3) = 0.15807381599999998	T(16) = 16.446766741
T(4) = 1.400296846	T(17) = 18.790904922
T(5) = 1.608827923	T(18) = 21.234707528
T(6) = 2.287241388	T(19) = 23.554405862
T(7) = 3.234931773	T(20) = 25.586081996
T(8) = 4.282501913	T(21) = 28.179043502999995
T(9) = 5.263174192	T(22) = 30.912973775999998
T(10) = 6.516431955	T(23) = 34.415719564999996
T(11) = 7.951758734	T(24) = 36.579495117
T(12) = 9.411626383	T(25) = 39.791264803
T(13) = 11.016058587	T(26) = 42.961423886

Complexity class = $\Theta(n^2)$

The graph plotted from the points calculated lined up much closer to $\Theta(n^2)$ than any of the other complexity classes, as shown in the graph above with the n^2 graph.

c)



Graph drawer: https://www.desmos.com/calculator

T(N/1000) = Mean/ 100_000

T(10) = 1.295179404	T(110) = 13.325634982
T(20) = 2.542758291	T(120) = 16.696805074
T(30) = 4.054791708000001	T(130) = 17.788283183
T(40) = 5.098346872	T(140) = 18.839631256
T(50) = 6.146665952	T(150) = 19.965950688
T(60) = 6.146665952	T(160) = 20.86227188
T(70) = 9.329933370000001	T(170) = 21.842347762000003
T(80) = 10.290874194	T(180) = 23.03065532
T(90) = 11.289025465	T(190) = 23.898224003
T(100) = 12.337431501	T(200) = 25.271648261000003

Complexity class = Θ (n)

For the optimised reapeatString() I concluded that the complexity class was Θ (n) as the dots plotted correlated to the straight line.