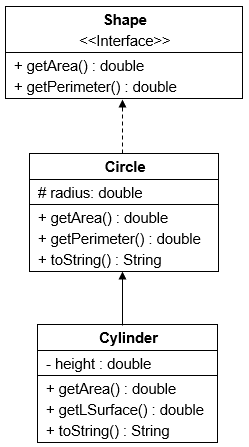
**Final Review**

**DATA STRUCTURES AND ALGORITHMS 1**

**Question 1:**

Based on the class diagram specified by UML (in the picture), implement the program in the Java programming language:



Where:

- For each derived class, you need to implement 3 constructors: parameterized constructor, non-parameterized constructor, and copy constructor.

- Method **getArea()** returns the area of the object corresponding to each class, which is calculated by the following formulas:

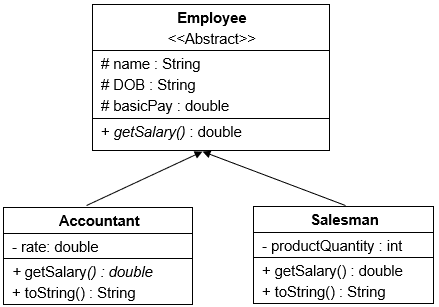
- Method**getPerimeter()**returns the perimeter of an object of class Circle:

- Method**getLSurface()** returns the volume of an object of classCylinder:

- Method**toString()** returns a string containing the attributes’ value of an object. For example, for the Cylinder class, the **toString()** method returns a string containing the radius and height.

**Question 2:**

Based on the class diagram specified by UML (in the picture), implement the program in the Java programming language:



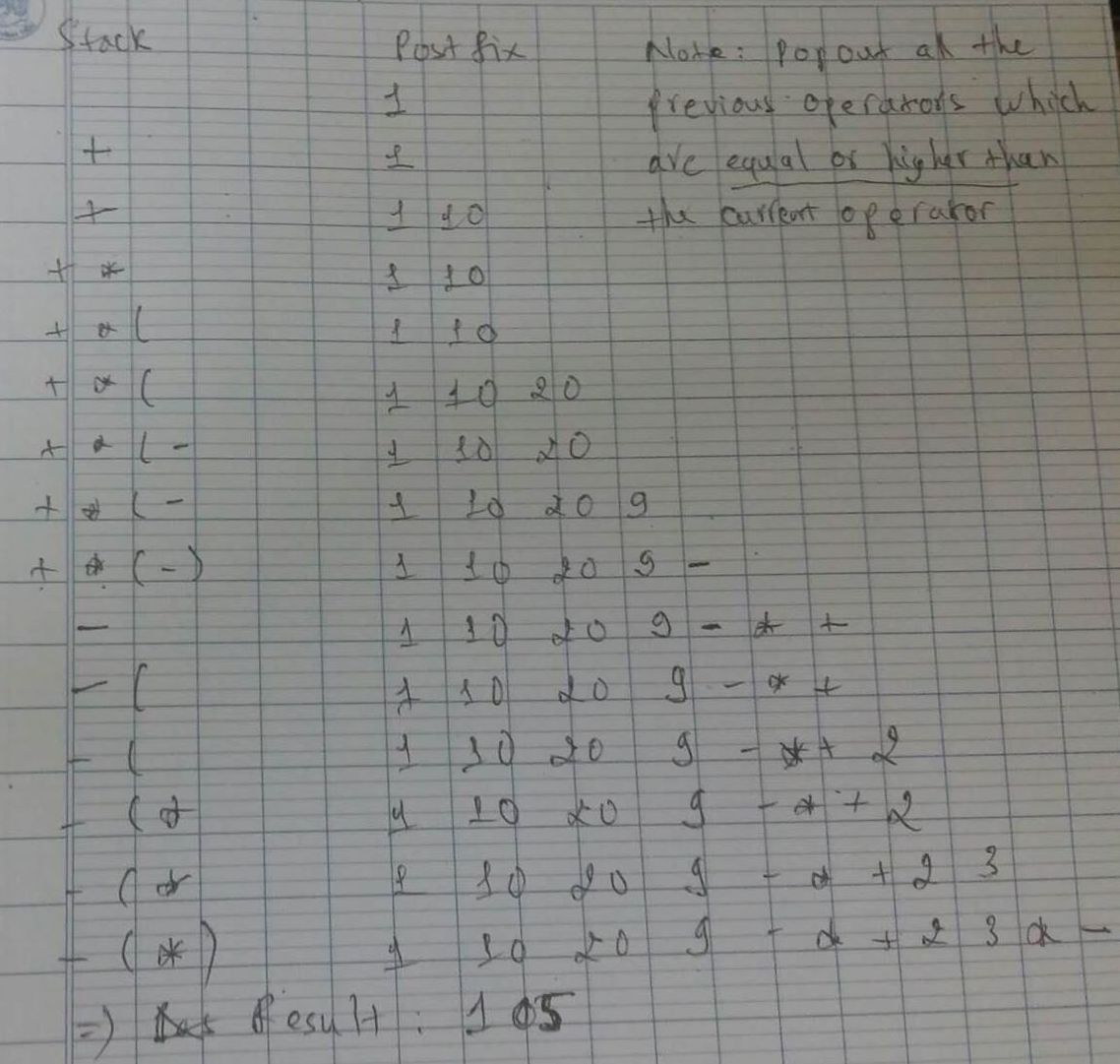
- For class Abstract Employee implement parameterized constructor and non-parameterized constructor. For class Accountant and Salesman, implement 3 constructors: parameterized constructor, non-parameterized constructor, and copy constructor.

- Method**getSalary()**returns the salary of the object corresponding to each class:

- Method**toString()** returns a string containing the attributes’ value of an object. For example, for classAccountant, the **toString()** method returns a string containing the name, DOB, basicPay and rate.

**Question 3:**Given an arithmetic expression:

**1 + 10 \* (20 - 9) - (2 \* 3)**

****

Convert the expression into postfix notation, then evaluate it using Reverse Polish algorithm

**Question 4:**Given an arithmetic expression:

**(10 \* 3 - 8) + (112 + 18\*213 - 199)**

****

Convert the expression into postfix notation, then evaluate it using Reverse Polish algorithm

**Question 5:**By recursive programming techniques, implement a method to calculate:

a.

b.

c. (giả sử n đượccholuônlớnhơn k)

**a/ public void An1(int n){**

**if(n==0)**

**return 4;**

**if(n==1)**

**return 3;**

**if(n>1)**

**return An1(n-1) - An1(n-2);**

**}**

**b/public void An2(int n){**

**if(n==1)**

**return 1;**

**if(n==2)**

**return 3;**

**if(n>2)**

**reutrn (An2(n-1)+An2(n-2))/2;**

**}**

**c/ public void An3(int n, int k){**

**if(k>n)**

**return 0;**

**if(k==0 || k==n)**

**return 1;**

**return An3(n-1,k) + An3(n-1,k-1);**

**}**

**Question6:**Given a list of integers:

31 1 20 0 99 12 27 2 36

Perform step by step **Bubble Sort** algorithm to arrange the list in ascending order

1 31 20 0 99 12 27 2 36

1 20 31 0 99 12 27 2 36

1 20 0 31 99 12 27 2 36

1 20 0 31 12 99 27 2 36

1 20 0 31 12 27 99 2 36

1 20 0 31 12 27 2 99 36

1 20 0 31 12 27 2 36 99

1 0 20 31 12 27 2 36 99

1 0 20 12 31 27 2 36 99

1 0 20 12 27 31 2 36 99

1 0 20 12 27 2 31 36 99

0 1 12 20 27 2 31 36 99

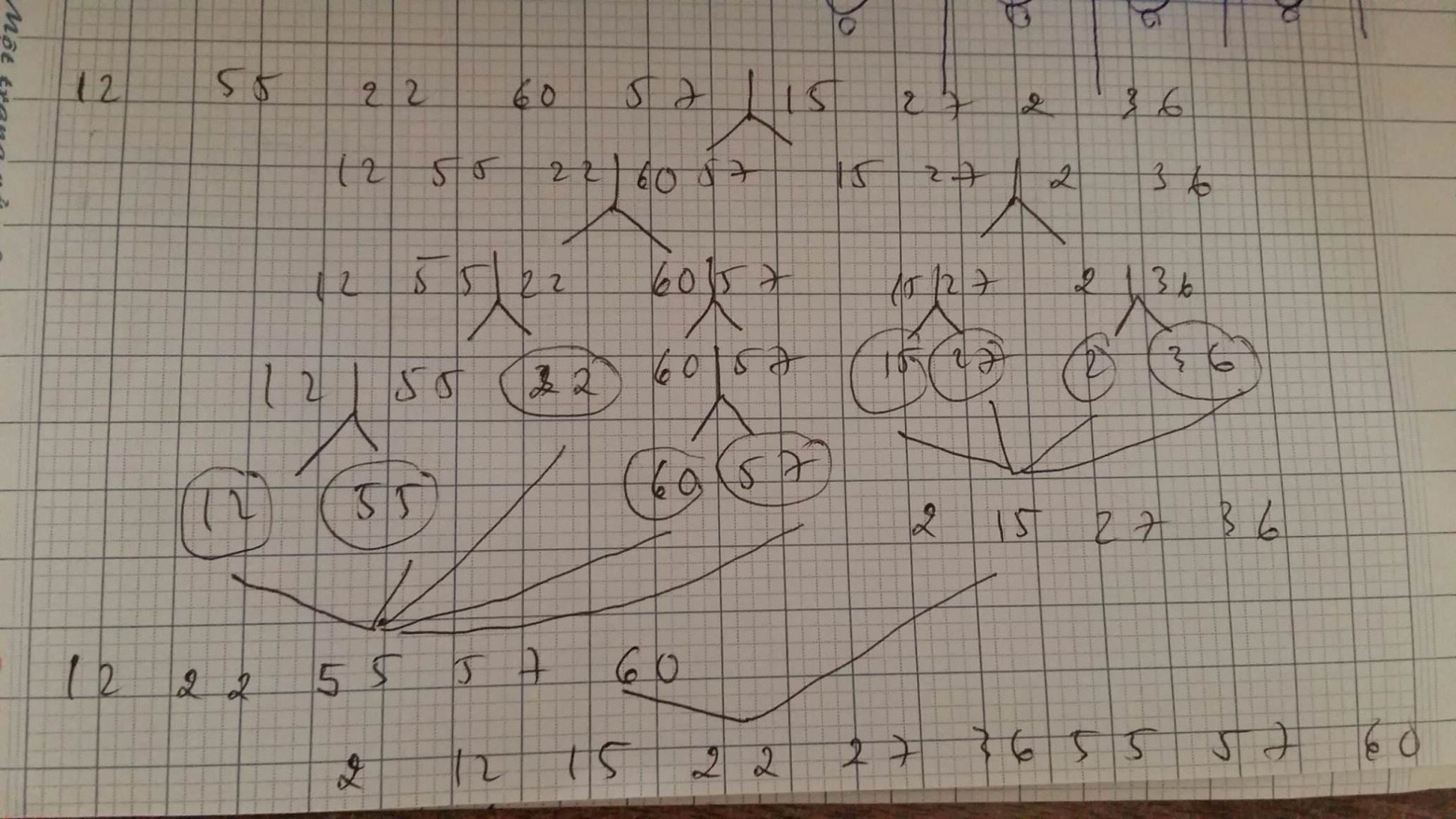
0 1 12 20 2 27 31 36 99

0 1 12 2 20 27 31 36 99

**Question7:**Given a list of integers:

12 55 22 60 57 15 27 2 36

Perform step by step **Merge Sort** algorithm to arrange the list in ascending order.

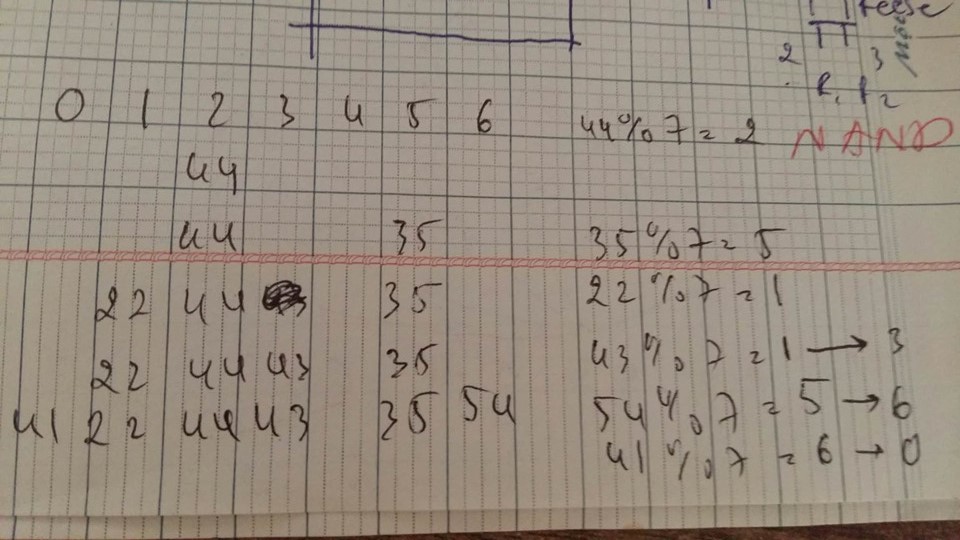


**Question8:**Calculate the complexity (Big-O) of the following programs (just show the result, don’t need to explain):

|  |  |
| --- | --- |
| **a.**  int i,j;  int s = 0;  for (i = 0; i < n; ++i)  {  for (j = 0; j < n; ++j)  {  s += j;  }  } => O(n^2) | **c.**  int s = 1;  for(i = 0; i < n; i++){  index = 1;  while(index < n){  s = s \* 3;  index \*= 5;  }  } => O(nlog(n)) |
| **b.**  for(i = 0; i < n; i++){  for(k = n\*n - 1; k >= 0; k--){  print("Hi");  }  } => O(n^3) | **d.**  int i, int j;  for(i = 0; i < n-1; i++) {  for(j = i+1; j < n; j++) {  tmp = A[i][j];  A[i][j] = A[j][i];  A[j][i] = tmp  }  } => O(n^2) |

**Question9:**Insert the following numbers to a hash table (size = 7), given ***Hash(k) = k mod 7***. In case of collision, apply **linear probing** (with step size of 1). Present the result in detail.

44 35 22 43 54 41



**Question10:**Insert the following numbers to a hash table (size = 13), given***Hash(k) = k mod 13***. In case of collision, apply **linear probing** (with step size of 1). Present the result in detail.

13 20 36232071312260283839

**-- END --**