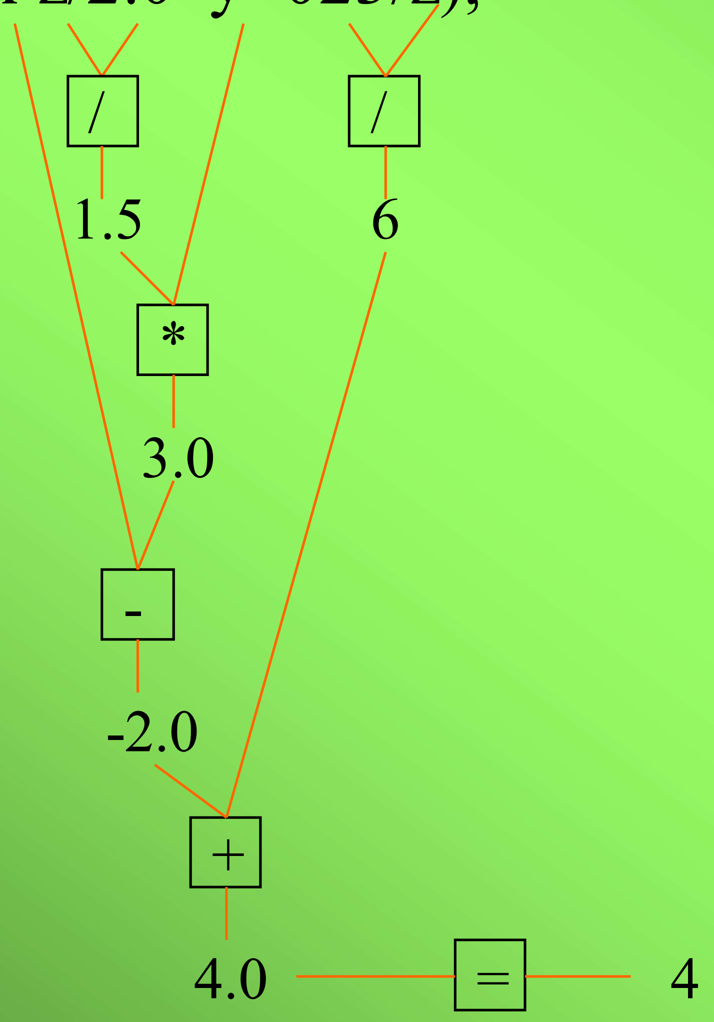
1(a)
```
if (--y == x) z += 3;
printf("%d %d %d\n", x, y, z);
```

1 1 6

1(b)
```
z = (int)(x-z/2.0*y+023/z);
```



1(c)
```
x -= y += z *= 2;
```

-7 8 6

1 2 4

```
2. for(i=1,j=0; i<=5; ++i) {
     switch(i) {
        case 1: j++;
        case 2: j += 3;
        case 3: j *= 2;
        case 4: break;
        case 5: continue;
     }
     System.out.printf("i=%d,j=%d\n", i, j);
  }
```

j $\boxed{\cancel{0}}$ $\cancel{1}$ $\cancel{4}$ $\cancel{8}$ $\cancel{11}$ $\cancel{22}$ 44

i=1,j=8
i=2,j=22
i=3,j=44
i=4,j=44

3. Show lines printed.

```
System.out.printf("%+6d,%-6d,%d\n", 321, 321, 321);
System.out.printf("%6d,%06d,%3c\n", 321, 321, 'a');
```

Col#	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
			+	3	2	1	,	3	2	1				,	3	2	1			
				3	2	1	,	0	0	0	3	2	1	,			a			

4.Design path-complete test cases for the program below;

```
while(x>0) {
      if (y<0)
            if (z<0)  {do something and then break; (1)}
            else {do something and then break; (2)}
      do something;
      x--; y--; z--;              x = 0,                    // 0 repetition
}                                 x = 1, y = -1, z = -1  // 1 repetition, exit break 1;
                                  x = 1, y = -1, z = 0   // 1 repetition, exit break 2;
                                  X=1, y = 0             // 1 repetition, exit from while(…);
                                  x = 2, y = 0, z=0      // 2 repetitions, exit from break 1;
                                  x = 2, y = 0, z=1      // 2 repetitions, exit from break 2;
                                  x=2, y =1              // 2 repetitions, exit from while(…)
```

```
package P;
public class A {
    int w;
    public int x;
    protected  int y;
    private int z;
}
```

```
import  P.*;
class D extends  A{
}
```

```
import  P.*;
class G extends  D{
}
```

```
package P;
public class B{
}
```

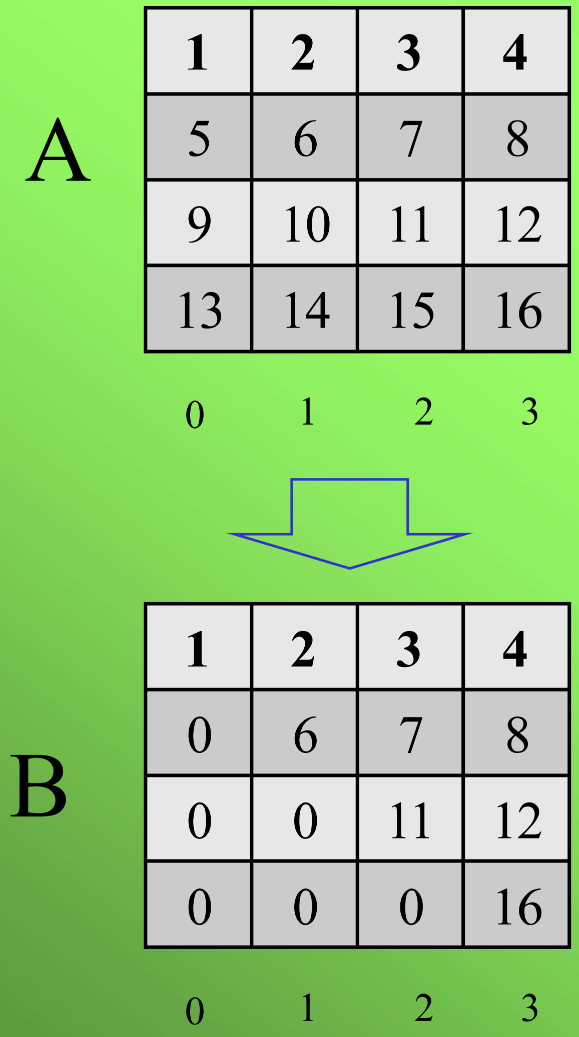
```
import  P.*;
class E extends  B{
}
```

```
package P;
public class  C{
}
```

```
import  P.*;
class F extends  C{
}
```

1. In what classes can w be accessed? A, B, C
2. In what classes can x be accessed? A, B, C, D, E, F, G
3. In what classes can y be accessed? A, B, C, D, G
4. In what classes can z be accessed? A

6. Design a Java method **upper(A)** to create and return a n×n array B such that the upper right corner of B contains copies of entries of the upper right corner of the argument n×n array A. (See the figure below.)



A

1	**2**	**3**	**4**
5	6	7	8
9	10	11	12
13	14	15	16
0	1	2	3

B

1	**2**	**3**	**4**
0	6	7	8
0	0	11	12
0	0	0	16
0	1	2	3

```
int[][] upper(int [][]A) {
        int n = A.length;
        int [][] B = new int[n][n];
        for (int i=0; i<n; ++i) {
                for(int j=i; j<n; ++j)
                        B[i] [j]= A[i][j];
        }
        return B;
}
```

6. Design a Java method lower(A) to create and return a n×n 2-D array B such that the lower left corner of B contains copies of entries of the lower left corner of A. (See the figure below.)

A

1	**2**	**3**	**4**
5	6	7	8
9	10	11	12
13	14	15	16
0	1	2	3

B

1	**0**	**0**	**0**
5	6	0	0
9	10	11	0
13	14	15	16
0	1	2	3

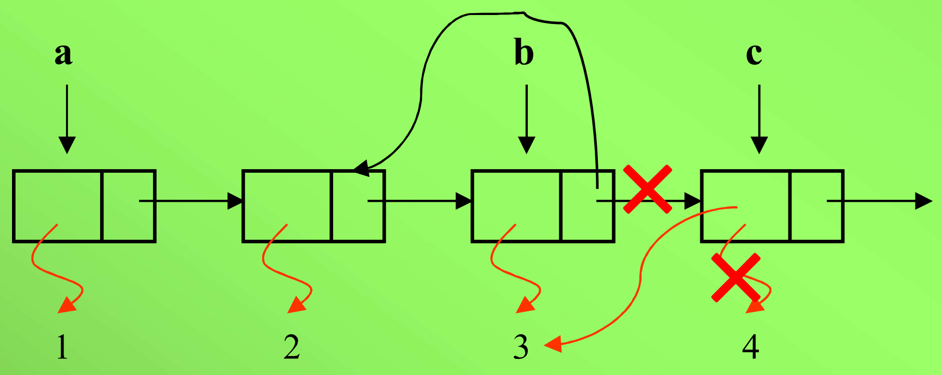
```
int[][] lower(int [][]A) {
        int n = A.length;
        int [][] B = new int[n][n];
        for (int i=0; i<n; ++i) {
                for(int j=0; j<=i; ++j)
                        B[i] [j]= A[i][j];
        }
        return B;
}
```

Part 2

1.

Assume that class Node is defined as in slide 17. Draw a picture to show the effect of the following statements.

```
b = a.Next.Next;
c = b.Next;
c.Data = b.Data;
b.Next = a.Next;
```



Use the algorithm given in class to convert the following infix expression to a postfix expression.

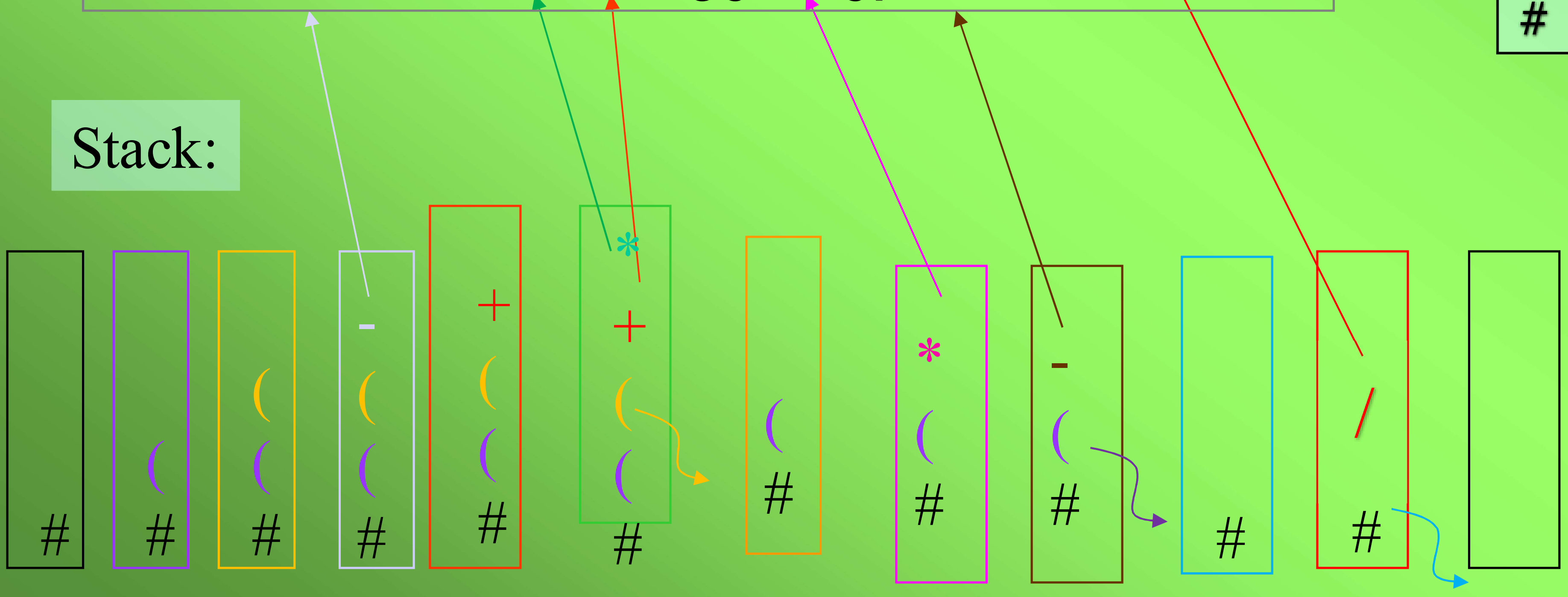
Infixt: ((12 - 23 + 34 * 45) * 56 – 67) / 89

Postfix:

12 23 - 34 45 * + 56 * 67 - 89 /

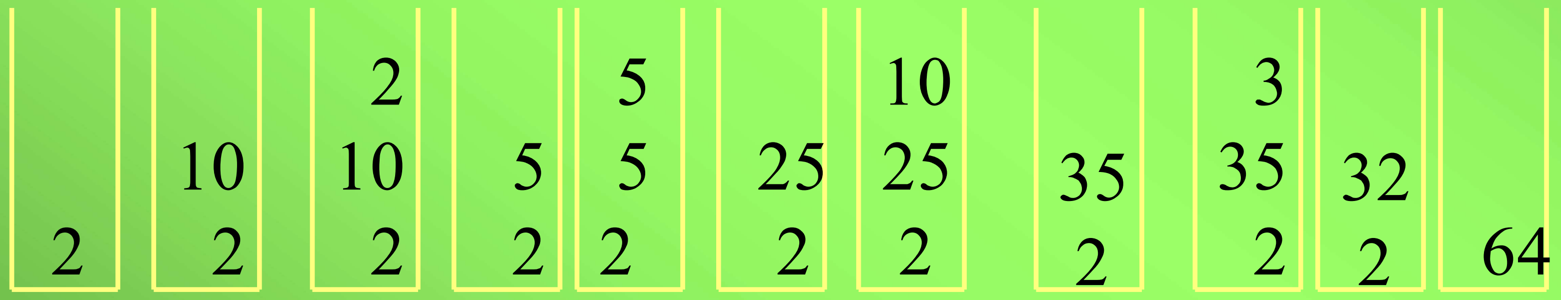
	Precedence
*,/	4
+,-	3
(	2
#	1

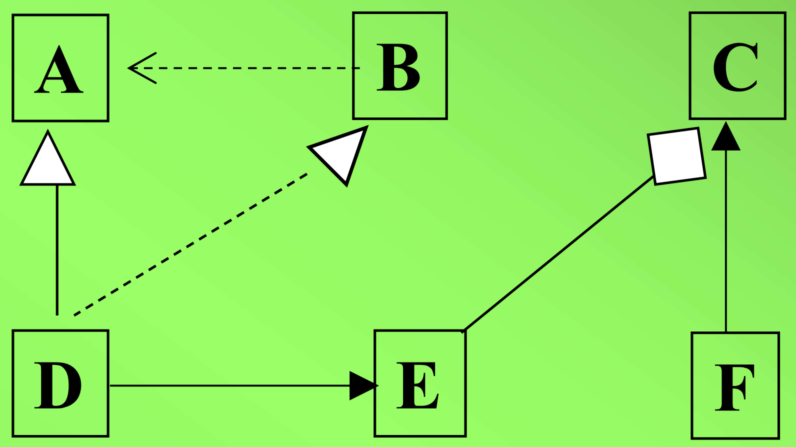
Stack:



Evaluate the following postfix expression.

2 10 2 / 5 * 10 + 3 - * -





In the following questions fill in is-a or has-a.

1. Class D _is-a___ class A.
2. Class D _has-a_ class E.
3. Class D _is-a____ class B
4. Class E _has-a_ class C.
5. Class F _has-a_ class C.

5. Add the following method to your SortedList class.

double average() – Returns the average of all numbers

```
double average() {
    DLNode cur = Head;
    int cnt = 0;
    double sum = 0.0;
    while(cur!=null) {
        sum += cur.Data;
        cnt++;
        cur = cur.Next;
    }
    return sum/cnt;
}
```