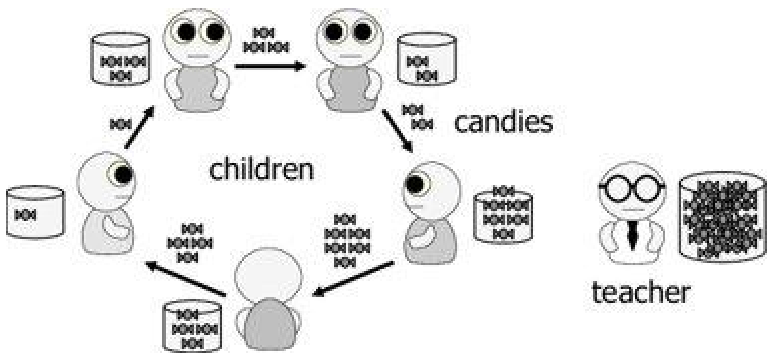
Competitive Programming Lab - 5

Academic year: 2020-2021 **Semester:** Long Sem

Faculty Name: Dr. Ajith Jublison sir **Date:** 25/ 6/ 2022

Student name: Taran Mamidala **Reg. no.:** 19BCE7346

Candies Distribution :



Q1.) There are N children standing in a line. Each child is assigned a rating value. You are giving candies to these children subjected to the following requirements:

1. Each child must have at least one candy,
2. Children with a higher rating get more candies than their neighbours. What are the minimum candies you must give?

Input Format:
The first and the only argument contains N integers in an array A.
Output Format
Return an integer, representing the minimum candies to be given.

Example:
Input 1:
A= [1, 2]
Output 1:
3

Explanation 1.
The candidate with 1 rating gets 1 candy and the candidate with rating cannot get 1 candy as 1 is its neighbour So rating 2 candidate gets 2 candies. In total, 2+1=3 candies need to be given out

Input 2

A [1, 5, 2, 1]

Output 2

7

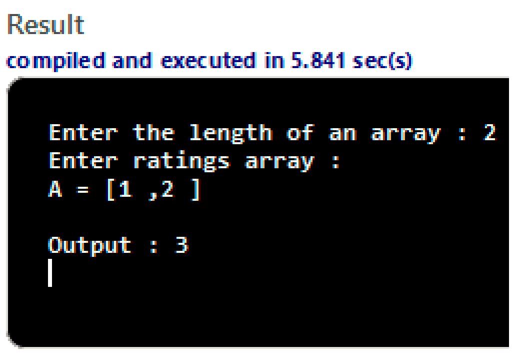
Explanation 2

Candies given = [1, 3, 2, 1]

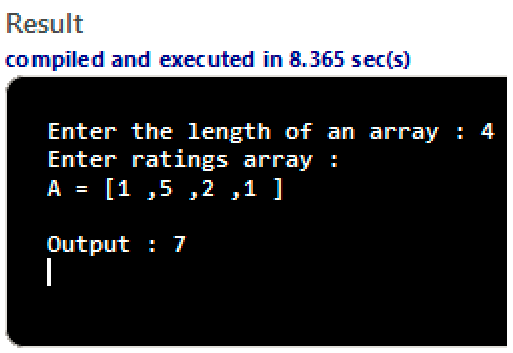
CODE:

```
import java.util.*;

public class CandiesDistribution {

      public int minimumCandies(int ratings[]) {

       boolean isIndependent = true;
      int maxHeight = 0;
      int differ = 0;
      int pointer = 0;
      int allCandies = 1;
      int presentCandy = 1;

      for (int i = 1; i < ratings.length; i++) {
            differ = 0;
            if (ratings[i] > ratings[i-1]) {
                  presentCandy += 1;
            }
            else if (ratings[i] == ratings[i-1]) {
                  isIndependent = true;
                  pointer = i;
                  presentCandy = 1;
            }
            else {
                  if (presentCandy == 1) {
                  if (!isIndependent) {
                        if (i - pointer == maxHeight - 1) {
                              pointer--;
                        }
                  }
                  }
                  else {
                  maxHeight = presentCandy;
                  presentCandy = 1;
                  isIndependent = false;
                  pointer = i;
                  }
                  differ = i - pointer;
            }
            allCandies += (differ + presentCandy);
      }
      return allCandies;
      }

      public static void main(String args[]) {

      Scanner sc = new Scanner(System.in);

      System.out.print("Enter the length of an array : ");
      int n=sc.nextInt();
      int[] rating = new int[n];
      System.out.print("\nEnter ratings array : \nA = [");
      rating[0] = sc.nextInt();
      for(int i=1; i<n;i++){
            System.out.print(",");
            rating[i] = sc.nextInt();
      }
      System.out.print("]\n");
      sc.close();

      CandiesDistribution candis = new CandiesDistribution();
  System.out.println("\nOutput : "+candis.minimumCandies(rating));
      }

}
```

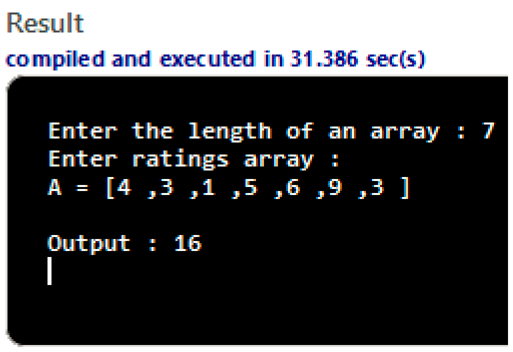
Output:


Result
compiled and executed in 5.841 sec(s)

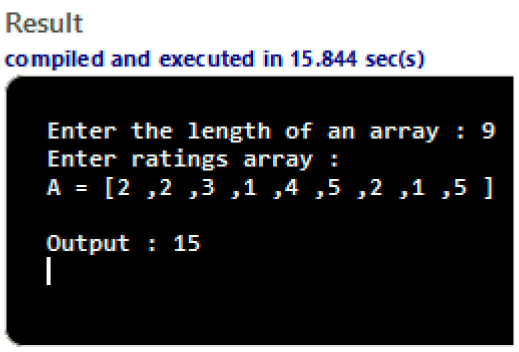
```
Enter the length of an array : 2
Enter ratings array :
A = [1 ,2 ]

Output : 3
```


Result
compiled and executed in 8.365 sec(s)

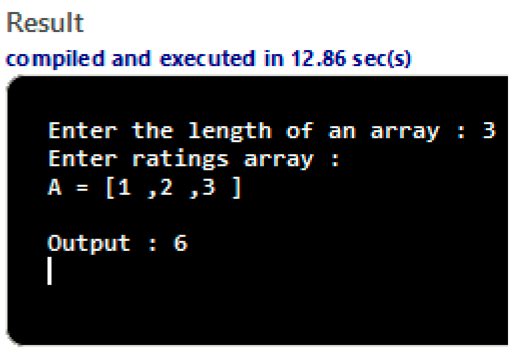
```
Enter the length of an array : 4
Enter ratings array :
A = [1 ,5 ,2 ,1 ]

Output : 7
```


Result
compiled and executed in 31.386 sec(s)

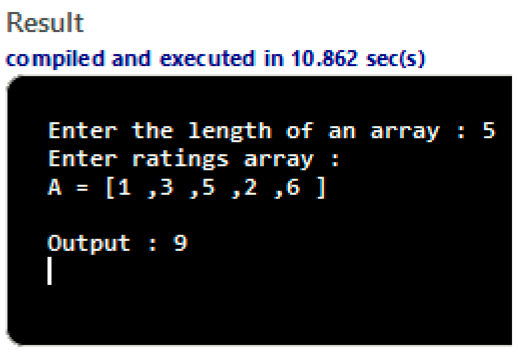
```
Enter the length of an array : 7
Enter ratings array :
A = [4 ,3 ,1 ,5 ,6 ,9 ,3 ]

Output : 16
```


Result
compiled and executed in 15.844 sec(s)

```
Enter the length of an array : 9
Enter ratings array :
A = [2 ,2 ,3 ,1 ,4 ,5 ,2 ,1 ,5 ]

Output : 15
```


Result
compiled and executed in 12.86 sec(s)

```
Enter the length of an array : 3
Enter ratings array :
A = [1 ,2 ,3 ]

Output : 6
```


Result
compiled and executed in 10.862 sec(s)

```
Enter the length of an array : 5
Enter ratings array :
A = [1 ,3 ,5 ,2 ,6 ]

Output : 9
```

Rabbit to Holes problem :

Q2.) There are N rabbit and N holes that are placed in a straight line. Each hole can accommodate only 1 mouse. The positions of rabbit are denoted by array A and the position of holes are denoted by array B.

A mouse can stay at his position, move one step right from x to x+1, or move one step left from x to x-1. Any of these moves consumes 1 minute. Assign rabbit to holes so that the time when the last mouse gets inside a hole is minimised.

Problem Constraints
1 <= N <- 105
-109<- A, B] <= 109
Input Format
First argument is an integer array A.
Second argument is an integer array B.

Output Format
Return an integer denoting the minimum time when the last rabbit gets inside the holes.

Example Input
Input 1
A-[-4,2,3]
B-[0,-2,41
Input 2:
A=[-2]
B=[-6]

Example Output

Output 1

2

Output 2

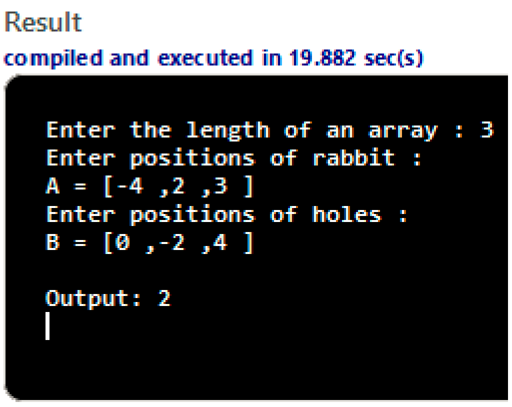
4

CODE:

```
import java.util.* ;

public class rabbitToHoles
{

      public int selectHole(ArrayList<Integer> rabbit,
ArrayList<Integer> holes)
      {
      if (rabbit.size() != holes.size())
            return -1;

      Collections.sort(rabbit);
      Collections.sort(holes);

      int size = rabbit.size();

      int maximium = 0;
      for (int i=0; i<size; i++)
            if (maximium < Math.abs(rabbit.get(i)-holes.get(i)))
                  maximium = Math.abs(rabbit.get(i)-holes.get(i));

      return Math.abs(maximium);
      }

      public static void main(String[] args)
      {
      Scanner sc= new Scanner(System.in);
      rabbitToHoles rth = new rabbitToHoles();
      System.out.print("Enter the length of an array : ");
      int n=sc.nextInt();
      ArrayList<Integer> rabbit = new ArrayList<Integer>();
      System.out.print("\nEnter positions of rabbit : \nA = [");
      rabbit.add(sc.nextInt());
      for(int i=1; i<n;i++){
```

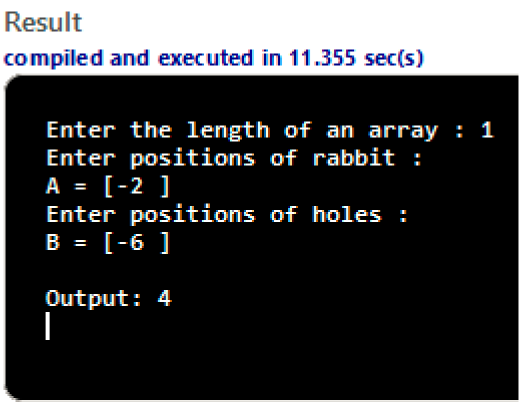
```
                System.out.print(",");
                int A = sc.nextInt();
                rabbit.add(A);
        }

        ArrayList<Integer> holes= new ArrayList<Integer>();
        System.out.print("]\nEnter positions of holes : \nB = [");
        holes.add(sc.nextInt());
        for(int i=1; i<n;i++){
                System.out.print(",");
                int B = sc.nextInt();
                holes.add(B);

        }

        System.out.println("]\n\nOutput: "+rth.selectHole(rabbit,
holes));
        }
}
```

Output:



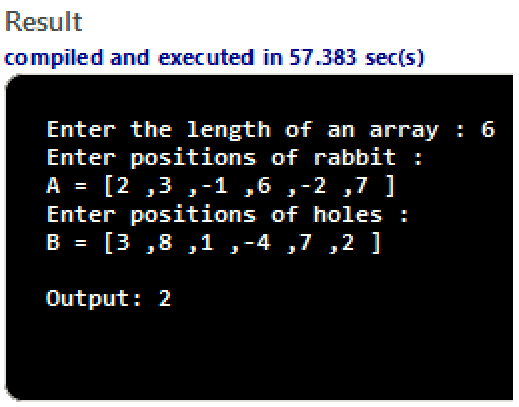
Result
compiled and executed in 19.882 sec(s)

```
Enter the length of an array : 3
Enter positions of rabbit :
A = [-4 ,2 ,3 ]
Enter positions of holes :
B = [0 ,-2 ,4 ]

Output: 2
```



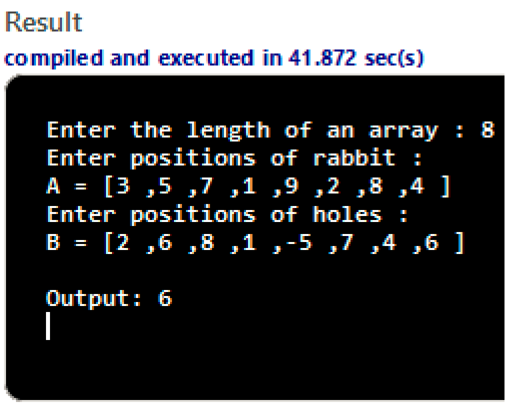
Result
compiled and executed in 11.355 sec(s)

```
Enter the length of an array : 1
Enter positions of rabbit :
A = [-2 ]
Enter positions of holes :
B = [-6 ]

Output: 4
```



Result
compiled and executed in 57.383 sec(s)

```
Enter the length of an array : 6
Enter positions of rabbit :
A = [2 ,3 ,-1 ,6 ,-2 ,7 ]
Enter positions of holes :
B = [3 ,8 ,1 ,-4 ,7 ,2 ]

Output: 2
```



Result
compiled and executed in 41.872 sec(s)

```
Enter the length of an array : 8
Enter positions of rabbit :
A = [3 ,5 ,7 ,1 ,9 ,2 ,8 ,4 ]
Enter positions of holes :
B = [2 ,6 ,8 ,1 ,-5 ,7 ,4 ,6 ]

Output: 6
```