

Given an integer x, return true if x is a palindrome, and false otherwise.

Note: Don't convert the integer to string.

Example 1:

Input: x = 121

Output: true

Example 2:

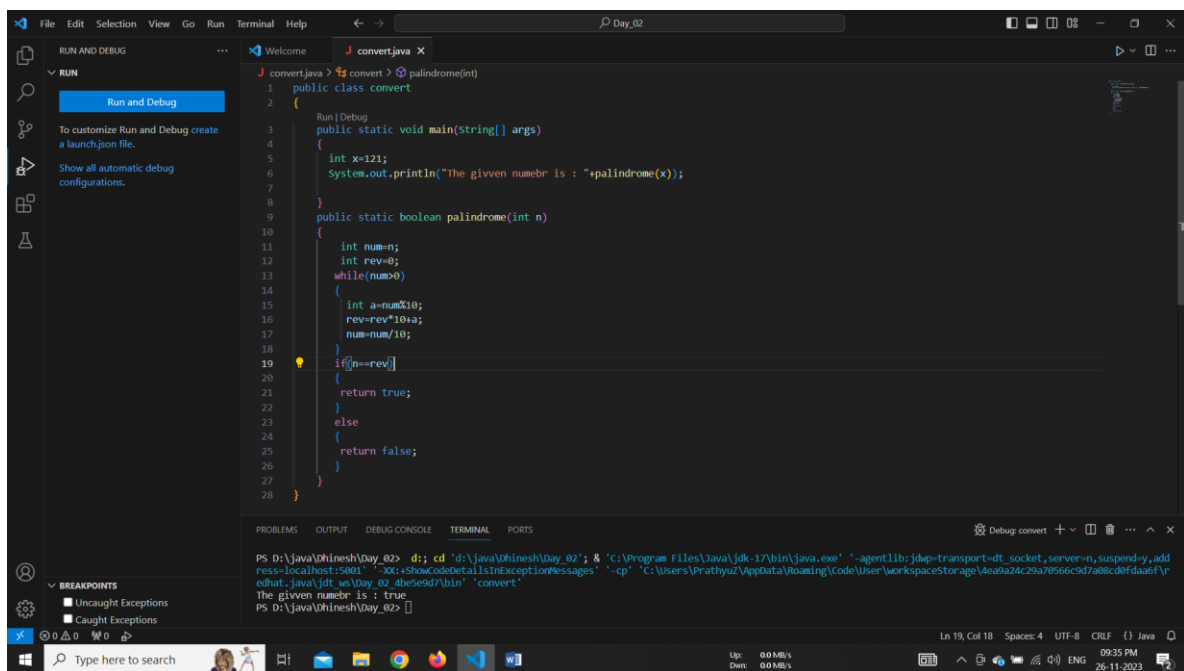
Input: x = -121

Output: false

Example 3:

Input: x = 10

Output: false



The screenshot shows an IDE with a Java file named 'convert.java'. The code defines a 'convert' class with a 'main' method and a 'palindrome' static method. The 'main' method initializes 'x' to 121 and prints the result of 'palindrome(x)'. The 'palindrome' method uses a loop to reverse the digits of 'n' and compares it with the original number. The terminal output shows the command to run the program and the output 'The given numbr is : true'.

```
1 public class convert
2 {
3     public static void main(String[] args)
4     {
5         int x=121;
6         System.out.println("The givne numebr is : "+palindrome(x));
7     }
8 }
9 public static boolean palindrome(int n)
10 {
11     int num=n;
12     int rev=0;
13     while(num>0)
14     {
15         int a=num%10;
16         rev=rev*10+a;
17         num=num/10;
18     }
19     if(n==rev)
20     {
21         return true;
22     }
23     else
24     {
25         return false;
26     }
27 }
28 }
```

```
PS D:\java\phinesh\Day_02> d; cd 'd:\java\phinesh\Day_02'; & 'c:\Program Files\Java\jdk-17\bin\java.exe' '-agentlib:jdwp-transport=dt_socket,server=n,suspend=y,add
ress=localhost:5001' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Prathyuz\AppData\Roaming\Code\User\workspaceStorage\4ea9a24c29a70566c3d7a88cdfdaa6f\
edhat-java\jdt-eclipse\Day_02_bin\bin' 'convert'
The given numbr is : true
PS D:\java\phinesh\Day_02>
```

To find Armstrong Number between two given number.

Example 1:

Input: start = 200, end = 500

Output: 370, 371, 407

Example 2:

Input: start = 1000, end = 2000

Output: 1634

Example 3:

Input: start = 1, end = 10

Output: 1, 2, 3, 4, 5, 6, 7, 8, 9

Given an integer num, repeatedly add all its digits until the result has only one digit, and return it. (use while loop only)

Example 1:

Input: num = 38

Output: 2

Explanation: The process is

38 --> 3 + 8 --> 11

11 --> 1 + 1 --> 2

Since 2 has only one digit, return it.

Example 2:

Input: num = 0

Output: 0

```
1 public class single {
2     public static void main(String[] args) {
3         int n=38;
4         int num=n;
5
6         while (num>=10)
7         {
8             int sum=0;
9             while (num>0)
10             {
11                 sum=sum+num%10;
12                 num=num/10;
13             }
14             num=sum;
15         }
16         int num = single.main(String[])
17     }
18     System.out.println("sum : "+num);
19 }
20
21
22
```

sum : 2

Finds the third-largest element present in the array.

Note: Don't use any sorting algorithms or build in sorting methods

Example 1 :

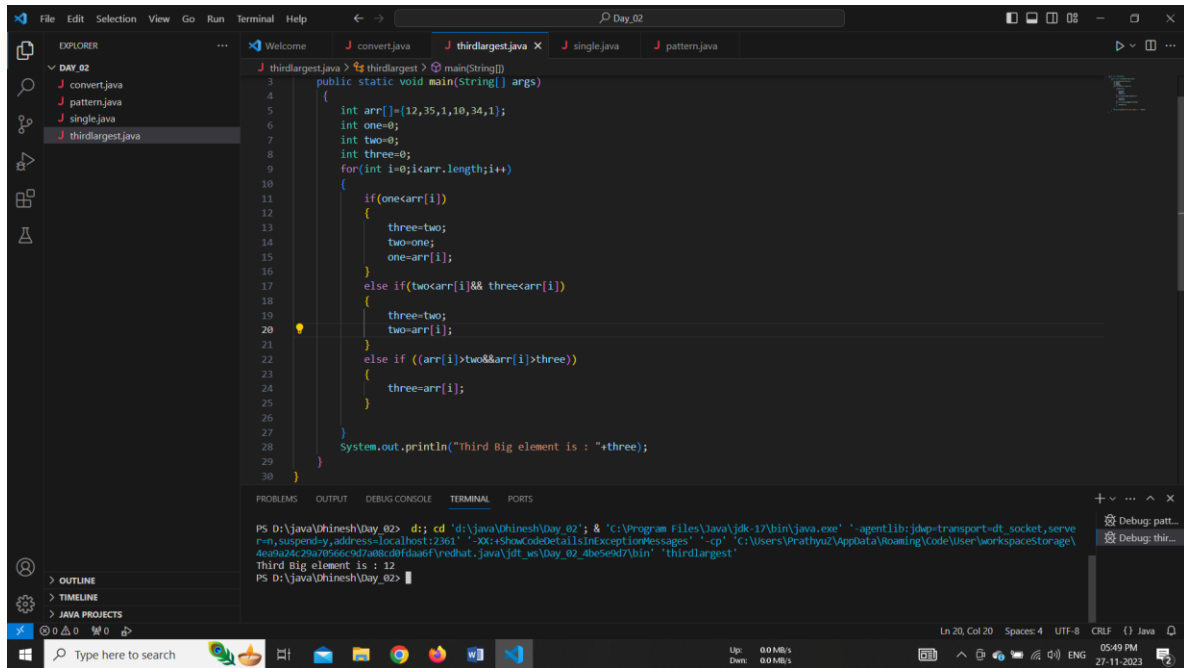
Input: arr[] = { 12, 35, 1, 10, 34, 1 }

Output: 12

Example 2:

Input: arr[] = { 2, 2, 2 }

Output: -1



Print the below pattern

```
*
**
***
****
*****
****
***
**
*
```

