Relative Speed

Problem statement

You are given a function,

```
int RelativeSpeed(int direction, int distance1, int time1, int distance2, int time2);
```

The function accepts integers 'direction', 'distance1', 'time1', 'distance2' & 'time2' as its argument. 'distance1' & 'distance2' are the distances travelled and 'time1' & 'time2' are the time taken by 1st & 2nd person respectively. Implement the function to find 'speed1' & 'speed2' of 1st & 2nd person respectively and return the relative speed as per given conditions:

- if direction = 0, relative speed = speed1 + speed2
- if direction = 1, relative speed = speed1 - speed2

Speed = Distance / Time

Assumption:

- speed1 >= speed2
- distance1, time1, distance2, time2 are greater than zero.

Note:

- direction is either 0 or 1.
- Computed value lies within integer range.
- Consider division as integer division.

Example:
Input:
direction: 0
distance1: 30
time1: 5
distance2: 20
time2: 10
Output:
8

Explanation:
speed1 = 30 / 5 = 6
speed2 = 20 / 10 = 2
Since direction = 0, relative speed = (6 + 2) = 8.

Sample Input

direction: 1
distance1: 50
time1: 2
distance2: 60
time2: 3

Sample Output

5

Instructions :

Solution :

```python
def RelativeSpeed(direction, distance1, time1, distance2, time2):
#Solved by Placement Solutions 2020-2021
    relativespeed = 0
    direction = int(input())
    distance1 = int(input())
    time1 = int(input())
    distance2 = int(input())
    time2 = int(input())
    if (direction==0) :
        result = (distance1/time1)+(distance2/time2)
    elif direction==1:
        result = (distance1/time1)-(distance2/time2)
    return result

result = RelativeSpeed(direction, distance1, time1, distance2, time2)
print(result)
```

```
1
50
2
60
3
5.0
```

Question 2 :

Problem statement

You are required to implement the following function:
char* FrequentCharacterReplaced(char* str, char x);

The function accepts a string 'str' and a character 'x' as its arguments. You are required to find the most frequent character in string 'str' and replace it with character 'x' across the string, and return the same.

Note:

- If frequency of two characters are same, we have to consider the character with lower ascii value.
- Character 'x' lies in the range [a - z]
- All characters in 'str' are in lower-case
- If 'str' is NULL, then return NULL. In case of Python, if 'str' is None, then return None.

Example:
Input:
str: bbadbbababb
x: t
Output:
ttadttatatt
Explanation:
The most frequent character in string 'str' is 'b', and replacing 'b' with 't' will form string 'ttadttatatt', hence 'ttadttatatt' is returned.

Sample Input
str: jjkdkksjjdjf
x: y

Sample Output
yykdkksyydyf

Instructions :

- This is a template based question, DO NOT write the "main" function.
- Your code is judged by an automated system, do not write any additional welcome/greeting messages.
- "Save and Test" only checks for basic test cases, more rigorous cases will be used to judge your code while scoring.
- Additional score will be given for writing optimized code both in terms of memory and execution time.

Now let's start coding :

| | | **C** | **C++** | **Java** | **C#** | **Python** |

> Read-only code below ...

```c
1  char* FrequentCharacterReplaced(char* str, char x);
2
3  int main() {
4      //Input read from STDIN
5      char* modstr = FrequentCharacterReplaced(str, x);
6      //Value in modstr is printed to STDOUT
7      return 0;
8  }
9
```

> Write your code below ...

```c
10 char* FrequentCharacterReplaced(char* str, char x) {
11     /* Write your code here. */
12 }
13
14
15
```

```python
1  def GetChar(myDict, k_max):
2      chList = []
3      for ch, val in myDict.items():
4          if(val == k_max):
5              chList.append(ch)
6      if(len(chList) == 1):
7          return chList.pop()
8      else:
9          aVal = [ord(ch) for ch in chList]
10         k = aVal.index(min(aVal))
11         return chList[k]
12 def FrequentCharacterReplaced(myStr, new_ch):
13     myDict = {}
14     chList = set(myStr)
15     for letter in chList:
16         myDict[letter] = myStr.count(letter)
17     k_max = max(myDict.values())
18     ch = GetChar(myDict, k_max)
19     return myStr.replace(ch, new_ch)
20 myStr, ch = input().split()
21 print(FrequentCharacterReplaced(myStr, ch))
```

```
jjkdkksjjdjf y
yykdkksyydyf
```

Question 3 :

You are required to implement the following function:
`int DifferenceSumOfDigits(int* arr, int n);`

The function accepts an array 'arr' of 'n' positive integers as its argument. Let's suppose:
$f(x)$ = Sum of digits of an integer

You are required to calculate the value of following:
$F_1 = [f(arr[0]) + f(arr[1]) + f(arr[2]) + \ldots + f(arr[n-1])] \% 10$

$F_2 = [(arr[0] + arr[1] + arr[2] + \ldots + arr[n-1])] \% 10$

$F = F_1 - F_2$

and return the value of F.

Note: $n > 0$

Example:
Input:
arr: 11 14 16 10 9 8 24 5 4 3
n: 10

Output:
-4

Explanation:
The value of F_1 is $(1 + 1) + (1 + 4) + (1 + 6) + (1 + 0) + (9) + (8) + (2 + 4) + (5) + (4) + (3)$ which is equal to 50 and (50 % 10) is 0 and value of F_2 is $[11 + 14 + 16 + 10 + 9 + 8 + 24 + 5 + 4 + 3]$ which is equal to 104 and (104 % 10) is 4, the value of F is (0 - 4), hence -4 is returned.

Sample Input
arr: 16 18 20
n: 3

Sample Output
4

Instructions :
- This is a template based question, DO NOT write the "main" function.
- Your code is judged by an automated system, do not write any additional welcome/greeting messages.
- "Save and Test" only checks for basic test cases, more rigorous cases will be used to judge your code while scoring.
- Additional score will be given for writing optimized code both in terms of memory and execution time.

Now let's start coding :

C C++ Java C# Python

> Read only code below

```
1 int DifferenceSumOfDigits(int* arr, int n);
2
3 int main() {
4     //Input read from STDIN
5     int p = DifferenceSumOfDigits(arr, n);
6     //Value in p is printed to STDOUT
7     return 0;
8 }
```

> Write your code below

```
10 int DifferenceSumOfDigits(int* arr, int n) {
11     /* Write your code here. */
12 }
```

Solution :

https://github.com/amirkhan1092

```
1  def GetSumOfDigit(num):
2      d_sum = 0;
3      while(num!=0):
4          d_sum = d_sum + (num%10)
5          num = num // 10
6      return d_sum
7  def DifferenceSumOfDigits(arr, n):
8      f1 = []
9      f2 = []
10     for i in range(n):
11         f1.append(arr[i])
12         f2.append(GetSumOfDigit(arr[i]))
13     return ((sum(f2)%10) - (sum(f1)%10))
14
15  n = int(input("Enter size: "))
16  arr = list(map(int, input("Enter elements: ").split()))
17  print(DifferenceSumOfDigits(arr, n))
```

```
Enter size: 10
Enter elements: 11 14 16 10 9 8 24 5 4 3
-4
```

Question 4 :

Problem statement

You are required to implement the following function:

```
int LargeSmallSum(int arr[], int length);
```

The function accepts an integer array 'arr' of size 'length' as its argument. You are required to find and return the sum of second large element from even position and second smallest element from odd position of given 'arr'.

Assumptions:

- All array elements are unique
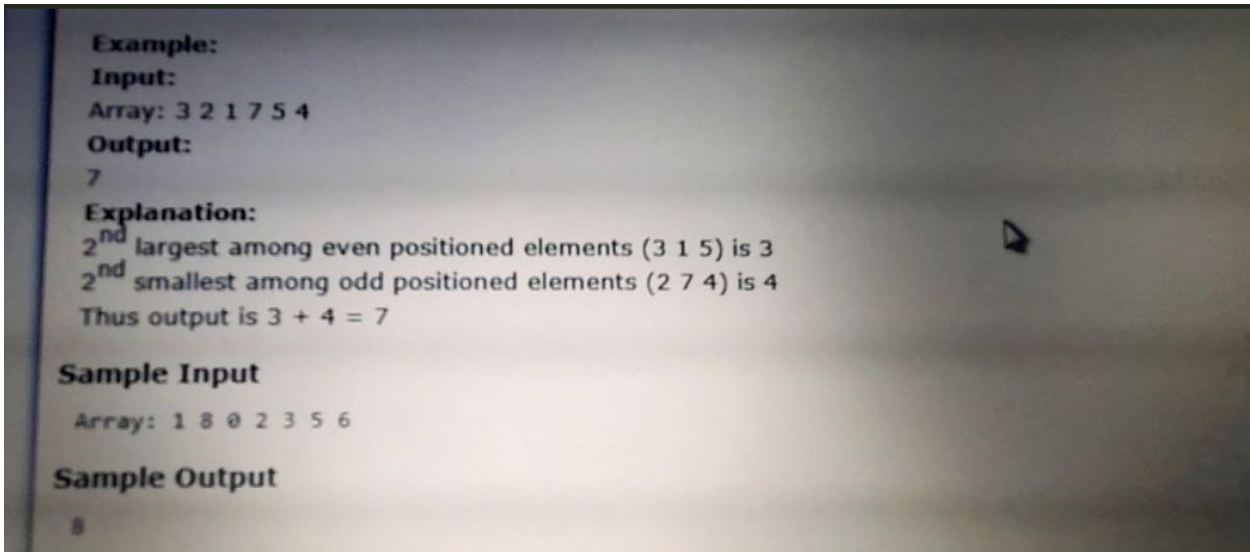- Treat the 0th index as even

Note:

- Return 0, if arr is empty.
- Return 0, if array length is 3 or less than 3.

Example:
Input:
Array: 3 2 1 7 5 4
Output:
7
Explanation:

Example:
Input:
Array: 3 2 1 7 5 4
Output:
7
Explanation:
2nd largest among even positioned elements (3 1 5) is 3
2nd smallest among odd positioned elements (2 7 4) is 4
Thus output is 3 + 4 = 7

Sample Input

Array: 1 8 0 2 3 5 6

Sample Output

8

Solution :

```python
def LargeSmallSum(arr, size):
    o = []
    e = []
    for i in range(size):
        if(i%2==0):
            e.append(arr[i])
        else:
            o.append(arr[i])
    e.sort()
    o.sort()
    return (o[1] + e[len(e)-2])

arr = list(map(int, input("Enter elements: ").split()))
if(len(arr)>3):
    print(LargeSmallSum(arr, len(arr)))
else:
    print(0)
```

```
Enter elements: 1 8 0 2 3 5 6
8
```