Basic Calculator

27 November 2024 00:05

```
 6 no of operands 6-1 number of operators
8 + 2 * 5 - 3 / 5 % 6
tokens = ["8", "+", "2", "*", "5", "-", "3", "/", "5", "%", "6"]


= 10 * 5 - 3 / 5 % 6
= 50 - 3 / 5 % 6
= 47 / 5 % 6
= 9 % 6
= 3

Read the input as a string
convert the input string into string array using split

store the first operand into result variable.
u can use Integer.parseInt(token[0])

iterate for loop from 1 to length of the string array: (step of 2)
       extract operator using charAt(0) to ch
       extract operand as Integer.parseInt(token[i+1]) to op

  use switch case for operator:
      for ex: case + : result=result+op
              case - : result=result-op
              case '/': result=result/op
               (in python) case '/':int(result/op)

             case % : result= (result%op+op) % op

    print(result)


For division in python
------------------------------

the behavior of the floor division operator (//) for negative numbers differs from other languages
like Java. In Python, // rounds
the result towards negative infinity, while in Java, integer division rounds towards zero.
Python behavior with //: -7 // 3 = -3 (rounded towards negative infinity).
Expected behavior (like in Java): -7 / 3 = -2 (rounded towards zero).
so floor division will not work if the number is negative.
instead we need to use type casting int(res/op)
*/


MOD OPERATOR BEHAVIOUR IN JAVA
-----------------------------------
For example: -7 % 5=-2 in java  but python gives 3.
but for negative number actual value is = 3 which is within range of 5
A remainder of -2 is mathematically valid but not in the typical range [0,5].
```

- In mathematics, the modulo operation often ensures that the result is non-negative and lies in the range [0, divisor) when the divisor is positive.
- This is useful in many practical scenarios, like indexing, cyclic arithmetic, or hashing, where results are expected to be non-negative.

Left rotation

27 November 2024 00:05

if array is 1,2,3 left rotation of array by 2 positions will give 3,1,2

if you want to left rotation of array by 5 positions still array will be 3,1,2

read array size
read the d the number of rotations
read the array elements

take d=d%n to avoid unnecessary rotations

loop from i=d to n and print array elements

loop from i=0 to d and print array elements

or use single for loop for i=0 to n
and print arr[(i+d)%n]

Right Rotation

27 November 2024 00:38

if array is 1,2,3 right rotation of array by 2 positions will give 2,3,1

if you want to left rotation of array by 5 positions still array will be 3,1,2

read array size
read the d the number of rotations
read the array elements

take d=d%n to avoid unnecessary rotations
calculate shift= n-d (last d elements come to front of array)

loop from i=n-d to n and print array elements

loop from i=0 to n-d and print array elements

or use single for loop for i=0 to n
and print arr[(i+shift)%n]

Array Key Segment Search

27 November 2024 00:49

```
input=
9
6 7 2 5 2 9 4 3 2
2
3
output=Yes the key found in every segment


Here key x is 2
segment size k is 3

{6,7,2}, {5,2,9},{4,3,2}

so key 2 is present in every segment

segmentkeysearch(int arr[], int n, int x, int k):
        if (k <= 0 || k > n)
            return

        for (int i = 0; i < n; i += k)
            int found = 0;
            for (int j = i; j < i + k && j < n; j++)
                if (arr[j] == x)
                    found = 1
                    break


            if (found == 0)
                return 0
```

```
for (int i = 0; i < n; i++) {
            if (arr[i] == x) {
                flag = 1;
            }
            if ((i + 1) % k == 0 || i == n - 1) {
                if (flag == 0) {
                    System.out.println(false);
                    return;
                }
                flag = 0;
            }
        }
        System.out.println(true);
```

Longest Consecutive 1's

05 November 2024 00:53

```
declare maxlength as 0  // to store the longest sequence of 1s
declare a as 0  // to count current sequence of contiguous 1s

for i from 0 to n - 1
    if arr[i] equals 1
        increment count
        if coubt greater than maxlength
            set maxlength to count

    if arr[i] equals 0
        set count to 0

print maxlength
```

Bubble sort efficient

27 November 2024 01:15

```
int temp;
boolean flag;
for (int i = 0; i < n - 1; i++) {
    flag = false;
    for (int j = 0; j < n - i - 1; j++) {
        if (arr[j] > arr[j + 1]) {

            temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
            flag = true;
        }
    }
    if (!flag) {
        break;
    }

    System.out.print("Pass " + (i + 1) + ": ");
    for (int k = 0; k < n; k++) {
        System.out.print(arr[k] + " ");
    }
    System.out.println();
}

System.out.println("Sorted array is:");
for (int i = 0; i < n; i++) {
    System.out.print(arr[i] + " ");
}
```

Lucky Numbers

27 November 2024 01:16

```
            {3, 7, 8}
            {9, 11, 13}
            {15, 16, 17}
```

here lucky number is 15 which is minimum in its row and maximum in its column

```
Initialize matrix with values
Create an empty list called luckyNumbers

Get number of rows m and columns n in the matrix

For i from 0 to m - 1
    Initialize minRowIndex to 0
    Initialize minRowValue to matrix[i][0]

    For j from 1 to n - 1
        If matrix[i][j] is < minRowValue
            Update minRowValue to matrix[i][j] and minRowIndex to j

    Initialize isMaxInColumn as true
    For k from 0 to m - 1
        If matrix[k][minRowIndex] is > minRowValue
            Set flag to false
            Break

    If flag is true
        Add minRowValue to luckyNumbers

Print luckyNumbers
```

String GCD

28 November 2024 14:27

Input
ABABAB
ABAB

Output
AB

```
        String P = sc.nextLine();
        String Q = sc.nextLine();
        if (!(P + Q).equals(Q + P)) {
            System.out.println(-1);
        } else {
            int gcdLength = gcd(P.length(),
Q.length());
            System.out.println(P.substring(0,
gcdLength));
        }
    }
    private static int gcd(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }
}
```

Spiral Traversal Matrix

27 November 2024 01:26

```
int[][] matrix = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};

int top = 0;
int bottom = matrix.length - 1;
int left = 0;
int right = matrix[0].length - 1;

while (top <= bottom && left <= right) {
    for (int i = left; i <= right; i++) {
        System.out.print(matrix[top][i] + " ");
    }
    top++;

     // Traverse from top to bottom along the right column
    for (int i = top; i <= bottom; i++) {
        System.out.print(matrix[i][right] + " ");
    }
    right--;


    if (top <= bottom) {
        // Traverse from right to left along the bottom row
        for (int i = right; i >= left; i--) {
            System.out.print(matrix[bottom][i] + " ");
        }
        bottom--;
    }

    if (left <= right) {
        // Traverse from bottom to top along the left column
        for (int i = bottom; i >= top; i--) {
            System.out.print(matrix[i][left] + " ");
        }
        left++;
    }
}
```

```
}
```

Zero Matrix

27 November 2024 01:39

```
Original Matrix:
 1 1 1
 1 0 1
 1 1 1
Modified Matrix:
 1 0 1
 0 0 0
 1 0 1
```

```
        boolean[] row = new boolean[n];
        boolean[] col = new boolean[n];
        // Step 1: Determine which rows and columns need to be zeroed
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (matrix[i][j] == 0) {
                    row[i] = true;
                    col[j] = true;
                }
            }
        }
        // Step 2: Set the rows to zero
        for (int i = 0; i < n; i++) {
            if (row[i]) {
                for (int j = 0; j < n; j++) {
                    matrix[i][j] = 0;
                }
            }
        }
        // Step 3: Set the columns to zero
        for (int j = 0; j < n; j++) {
            if (col[j]) {
                for (int i = 0; i < n; i++) {
                    matrix[i][j] = 0;
                }
            }
        }
    }
```

```
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
```

```
}
```

Smaller<X<Greater

27 November 2024 00:58

```
input=10
2 1 3 4 5 7 12 45 78 10
output=
Index of the element: 2


Approach
-------------

intitalize index to -1

start loop from i=1 to n-1:
    take 2 variables
     left=i-1
     right=i+1
     flag=true

     while (left >=0)
           if (a[left]>a[i])
             flag=false
             break
      left--
      end while loop

      while (right < n)

        if (a[Right] < a[i])
                  flag = false
                    break
      right++
      end while loop

      if flag=true
        index will be i
        break for loop


      if index!=-1
         print the index of element: index
```

```
else
   print no such element exists
```

```
int n = scanner.nextInt();
int[] arr = new int[n];


for (int i = 0; i < n; i++) {
    arr[i] = scanner.nextInt()
}

int index = -1

for (int i = 1; i < n - 1; i++)
    int Left = i - 1
    int Right = i + 1
    boolean flag = true

    // Check all elements before the current element
    while (Left >= 0) {
        if (arr[Left] > arr[i]) {
            flag = false;
            break;
        }
        Left--


    // Check all elements after the current element
    while (Right < n)
        if (arr[Right] < arr[i]) {
            flag = false
            break
        }
        Right++



    if (flag==true)
        index = i
        break
```

```
if (index != -1)
    System.out.println("Index of the element: " + index)
 else
    System.out.println("No such element found")
```

XoR of sum of all Pairs

27 November 2024 01:50

if array is [1,2,3]
Pairs and their sums:

- (1 + 1) = 2
- (1 + 2) = 3
- (1 + 3) = 4
- (2 + 1) = 3
- (2 + 2) = 4
- (2 + 3) = 5
- (3 + 1) = 4
- (3 + 2) = 5
- (3 + 3) = 6

XOR of these sums:
Start with result = 0.

- result ^= 2 → result = 0 ^ 2 = 2
- result ^= 3 → result = 2 ^ 3 = 1
- result ^= 4 → result = 1 ^ 4 = 5
- result ^= 3 → result = 5 ^ 3 = 6
- result ^= 4 → result = 6 ^ 4 = 2
- result ^= 5 → result = 2 ^ 5 = 7
- result ^= 4 → result = 7 ^ 4 = 3
- result ^= 5 → result = 3 ^ 5 = 6
- result ^= 6 → result = 6 ^ 6 = 0

```
        int[] arr = {1, 2, 3};
        int n = arr.length;
        int result = 0;


        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                int sum = arr[i] + arr[j];
                result ^= sum;
            }
        }

        System.out.println("XOR of sums of all pairs: " + result);
```

An Efficient approach is based upon the fact that xor of the same values is 0. All the pairs like (a[i], a[j]) and (a[j], a[i]) will have same sum. So, their xor values will be 0. Only the pairs like (a[i], a[i]) will give the different result. So, take the xor of all the elements of the given array and multiply it by 2.

```
 int xoR = 0;
        for (int i = 0; i < n; i++) {
            xoR = xoR ^ arr[i]
        }
```

```
    print xoR * 2
}
```

Sum of XoR of all pairs

27 November 2024 01:51

```
Input : arr[] = {5, 9, 7, 6}
Output : 47

5 ^ 9 = 12

9 ^ 7 = 14

7 ^ 6 = 1

5 ^ 7 = 2

5 ^ 6 = 3

9 ^ 6 = 15

Sum = 12 + 14 + 1 + 2 + 3 + 15

    = 47
```

```
        int[] arr = {5, 9, 7, 6};
        int n = arr.length;
        int result = 0;

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                int xorSum = arr[i] ^ arr[j];
                result += xorSum;
            }
        }

        System.out.println("XOR sum of all pairs: " + result);
```