**RUBRIC & Solution**

**SET-A**

|  | **Criteria** | **Marks** |
| --- | --- | --- |
| **1** | **Calculating the length of the stack** | **3** |
| **2** | **Store odd and even elements in temp stack** | **4** |
| **3** | **Handle both scenarios** | **5** |
| **4** | **Push the integers correctly in the original stack** | **2** |
| **5** | **Returning the original stack** | **1** |

**Solution:**

| **Python** | **JAVA** |
| --- | --- |
| def rearrange\_stack(input\_stack):  odd\_stack = Stack()  even\_stack = Stack()  len\_count = 0  while not input\_stack.isEmpty():  num = input\_stack.pop()  len\_count += 1  if num % 2 == 0:  even\_stack.push(num)  else:  odd\_stack.push(num)  #Scenario 1  if len\_count % 2 == 0:  while not odd\_stack.isEmpty() or not even\_stack.isEmpty():  if not even\_stack.isEmpty():  input\_stack.push(even\_stack.pop())  if not odd\_stack.isEmpty():  input\_stack.push(odd\_stack.pop())  #Scenario 2  else:  while not odd\_stack.isEmpty() or not even\_stack.isEmpty():  if not odd\_stack.isEmpty():  input\_stack.push(odd\_stack.pop())  if not even\_stack.isEmpty():  input\_stack.push(even\_stack.pop())  return input\_stack | public static void rearrangeStack(Stack inputStack) {  Stack oddStack = new Stack();  Stack evenStack = new Stack();  int lenCount = 0;  // First pass: separate odd and even numbers  while (!inputStack.isEmpty()) {  int num = inputStack.pop();  lenCount++;    if (num % 2 == 0) {  evenStack.push(num);  } else {  oddStack.push(num);  }  }  // Scenario 1:  if (lenCount % 2 == 0) {  while (!oddStack.isEmpty() || !evenStack.isEmpty()) {  if (!evenStack.isEmpty()) {  inputStack.push(evenStack.pop());  }  if (!oddStack.isEmpty()) {  inputStack.push(oddStack.pop());  }  }  }  // Scenario 2:  else {  while (!oddStack.isEmpty() || !evenStack.isEmpty()) {  if (!oddStack.isEmpty()) {  inputStack.push(oddStack.pop());  }  if (!evenStack.isEmpty()) {  inputStack.push(evenStack.pop());  }  }  }  return inputStack;  } |

**SET-B**

|  | **Criteria** | **Marks** |
| --- | --- | --- |
| **1** | **Store odd and even elements in temp stack** | **4** |
| **2** | **Calculating the summation of the odd and even elements** | **3** |
| **3** | **Handling the push condition** | **5** |
| **4** | **Push the integers in the original stack** | **2** |
| **5** | **Returning the original stack** | **1** |

**Solution:**

| **Python** | **JAVA** |
| --- | --- |
| def rearrange\_stack(input\_stack):  odd\_stack = Stack()  even\_stack = Stack()  odd\_sum = 0  even\_sum = 0    while not input\_stack.isEmpty():  num = input\_stack.pop()  if num % 2 == 0:  even\_stack.push(num)  even\_sum += num  else:  odd\_stack.push(num)  odd\_sum += num    if even\_sum < odd\_sum:  while not even\_stack.isEmpty():  input\_stack.push(even\_stack.pop())  while not odd\_stack.isEmpty():  input\_stack.push(odd\_stack.pop())  else:  while not odd\_stack.isEmpty():  input\_stack.push(odd\_stack.pop())  while not even\_stack.isEmpty():  input\_stack.push(even\_stack.pop())  return input\_stack | public static Stack rearrangeStack(Stack inputStack) {  Stack oddStack = new Stack();  Stack evenStack = new Stack();  int oddSum = 0;  int evenSum = 0;    // Separate numbers into odd and even stacks and calculate their sums  while (!inputStack.isEmpty()) {  int num = inputStack.pop();  if (num % 2 == 0) {  evenStack.push(num);  evenSum += num;  } else {  oddStack.push(num);  oddSum += num;  }  }    // Rearrange based on sum comparison  if (evenSum < oddSum) {  // Push even stack first, then odd stack  while (!evenStack.isEmpty()) {  inputStack.push(evenStack.pop());  }  while (!oddStack.isEmpty()) {  inputStack.push(oddStack.pop());  }  } else {  // Push odd stack first, then even stack  while (!oddStack.isEmpty()) {  inputStack.push(oddStack.pop());  }  while (!evenStack.isEmpty()) {  inputStack.push(evenStack.pop());  }  }    return inputStack;  } |

Note\*: There are multiple ways to solve these problem, and appropriate marks can be given for each approach based on its correctness and efficiency.