**Set A**

| **Python** | **Java** |
| --- | --- |
| def rotate\_stack(st, k):  # get the length of the stack  len = 0  tempStack = Stack()  while not st.isEmpty():  tempStack.push(st.pop())  len += 1  # get the effective rotation  k = k % len  # rotate the stack  tempStack2 = Stack()  for i in range(len - k):  tempStack2.push(tempStack.pop())    for i in range(k):  st.push(tempStack.pop())    for i in range(len - k):  tempStack.push(tempStack2.pop())    for i in range(len - k):  st.push(tempStack.pop())  return st | static Stack rotate\_stack(Stack st, int k){  // get the length of the stack  int len = 0;  Stack temp = new Stack();  while(!st.isEmpty()){  temp.push(st.pop());  len++;  }  // get the effective rotation  k = k % len;  // rotate the stack  Stack temp2 = new Stack();  for(int i = 0; i < len - k; i++){  temp2.push(temp.pop());  }  for(int i = 0; i < k; i++){  st.push(temp.pop());  }  for(int i = 0; i < len - k; i++){  temp.push(temp2.pop());  }  for(int i = 0; i < len - k; i++){  st.push(temp.pop());  }  return st;  } |

**Set B**

| **Python** | **Java** |
| --- | --- |
| #rotate downwards  def rotate\_stack(st, k):  # get the length of the stack  len = 0  tempStack = Stack()  while not st.isEmpty():  tempStack.push(st.pop())  len += 1  # get the effective rotation  k = k % len  # rotate the stack  tempStack2 = Stack()  for i in range(k):  tempStack2.push(tempStack.pop())    for i in range(len - k):  st.push(tempStack.pop())    for i in range(k):  tempStack.push(tempStack2.pop())    for i in range(k):  st.push(tempStack.pop())    return st | static Stack rotate\_stack(Stack st, int k){  // get the length of the stack  int len = 0;  Stack temp = new Stack();  while(!st.isEmpty()){  temp.push(st.pop());  len++;  }  // get the effective rotation  k = k % len;  // rotate the stack  Stack temp2 = new Stack();  for(int i = 0; i < k; i++){  temp2.push(temp.pop());  }  for(int i = 0; i < len - k; i++){  st.push(temp.pop());  }  for(int i = 0; i < k; i++){  temp.push(temp2.pop());  }  for(int i = 0; i < k; i++){  st.push(temp.pop());  }  return st;  } |

**RUBRIC**

|  | **Criteria** | **Marks** |
| --- | --- | --- |
| **1** | **Properly declaring method/function using proper parameter** | **1** |
| **2** | **Calculating the length of the stack** | **2** |
| **3** | **Calculate the effective rotation** | **1** |
| **4** | **Store top (len - k) [Set A] or (k) [Set B] elements in another stack** | **3** |
| **5** | **Push the remaining into the original Stack** | **3** |
| **6** | **Previously Stored elements need reversal, use another stack for that** | **2** |
| **7** | **Push back all the remaining elements into original stack** | **2** |
| **8** | **Return the original stack** | **1** |

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