**Set A**

| **Python** | **Java** |
| --- | --- |
| def conditional\_reverse(st,n):  Stack1 = Stack()  Stack2 = Stack()  for i in range(n):  Stack1.push(st.pop())  while not Stack1.isEmpty():  Stack2.push(Stack1.pop())  while not Stack2.isEmpty():  st.push(Stack2.pop())  return st | public static Stack conditional\_reverse(Stack st, int n) {  Stack Stack1 = new Stack();  Stack Stack2 = new Stack();  for (int i = 0; i < n; i++) {  Stack1.push(st.pop());  }  while (!Stack1.isEmpty()) {  Stack2.push(Stack1.pop());  }  while (!Stack2.isEmpty()) {  st.push(Stack2.pop());  }  return st;  } |

**Rubric**

|  | **Criteria** | **Marks** |
| --- | --- | --- |
| 1 | Properly declaring method/function with proper parameter (st, n) | 1 |
| 2 | Initializing two auxiliary stack | 2 |
| 3 | Loop to transfer n elements from st to Stack1 | 4 |
| 4 | Loop to transfer all elements from Stack1 to Stack2 | 4 |
| 5 | Loop to transfer all elements from Stack2 back to st | 3 |
| 6 | Returning the modified stack | 1 |
|  | **Total** | **15** |

**Set B**

| **Python** | **Java** |
| --- | --- |
| def conditional\_reverse(st,n):  Stack1 = Stack()  Stack2 = Stack()  count = 0  while not st.isEmpty():  Stack1.push(st.pop())  count += 1  for i in range(count-n):  Stack2.push(Stack1.pop())  while not Stack2.isEmpty():  st.push(Stack2.pop())  while not Stack1.isEmpty():  st.push(Stack1.pop())  return st | public static Stack conditional\_reverse(Stack st, int n) {  Stack Stack1 = new Stack();  Stack Stack2 = new Stack();  int count = 0;  while (!st.isEmpty()) {  Stack1.push(st.pop());  count++;  }  for (int i = 0; i < count - n; i++) {  Stack2.push(Stack1.pop());  }  while (!Stack2.isEmpty()) {  st.push(Stack2.pop());  }  while (!Stack1.isEmpty()) {  st.push(Stack1.pop());  }  return st;  } |

**Rubric**

|  | **Criteria** | **Marks** |
| --- | --- | --- |
| 1 | Properly declaring method/function with proper parameter (st, n) | 1 |
| 2 | Initializing two auxiliary stack | 2 |
| 3 | Loop to transfer all elements from st to Stack1 while maintaining a count | 2 |
| 4 | Loop to transfer the first count-n elements from Stack1 to Stack2 | 3 |
| 5 | Loop to transfer all elements from Stack2 back to st | 3 |
| 6 | Loop to transfer the remaining elements in Stack1 back to st, ensuring the top n elements are reversed | 3 |
| 7 | Returning the modified stack | 1 |
|  | **Total** | **15** |