Hands-On 4

Aidan Mondress
1001841887

```
int merge()
{
  int k = 3;
  int N=4;
  int inputArr[k][N] = {...};

  int mergedArr [k*N]
  for(int i =0; i<k;i++)          $\sum_{i=0}^{k} 1$
  {
    for(int j =0; j<N;j++)              $\sum_{int\ i=0}^{k} \sum_{j=0}^{N} 1$
      mergedArr[] = inputArr[i][j]      $\sum_{int\ i=0}^{k} \sum_{j=0}^{N} 1$
  }
  insertionSort(mergedArr);

  Printf("Merged Array: ");
  printArr(mergedArr);

  return 0;
}
```

Total= $k + nk + nk = 2nk + k = O(nk)$

OPTIMIZE:

For this purpose, I feel that I should have used selection sort instead of insertion sort, for a more efficient algorithm.

```
int removeDupes(int arr[], int n)
{
  if(n ==1 || n ==0)
     return n;
  int temp[n]
  int count = 0;
  for(int i=0;i<n-1;i++)
  {
    if(arr[i] != arr[i+1])
       temp[count++] = arr[i];
  }
  temp[count++] = arr[n-1];
  for (int i=0; i<count;i++)
     arr[i] = temp[i];
  return count;
}
```

$\sum_{i=0}^{n-1} 1$

$\sum_{i=0}^{n-2} 1$

$\sum_{i=0}^{n-2} 1$

Total: $n-1 + n-2 + n-2 = 3n-5 = O(n)$

OPTIMIZE:

We could optimize this array by directly removing duplicates from the array instead of constructing an additional temp array.