Rohan Javed Assignment 2

L21-5625
BSDS-5B

—(Q1)—

Sorting Colours

```
Sort Colours(Arr[]){
  int i=0
  int j=i+1
  ke = leng[Arr]-1
while (j ≤ k)
  if (Arr[i]==Red)
    {
      swap Arr(i], Arr(j]
      i++
      j++
    }
  elseif (Arr(i] == white)
    { j++ }

  else
    {
      swap Arr(k] , Arr(j]
      j++
      k--  }

    }
  return Arr.

}
```

(Q 2)

```
mergesoft (X(), Y[], new[ ], m, N)
    from (1 : N)
      left = 1
      right = m
      while (right > left)
       { mid = (right + left)/2
                    }
 if  X[mid] ≤ Y[i] and [mid = m || X[m+1]
     > Y[i]).
     return 0;

 else if   X[mid] ≤ Y[i]
           left = mid + 1;

  else
       right = mid - 1

 for  j (m : mid+1)
    X [(i+1) = X [j]
 X[mid+1] = Y[i]

m = m + 1

 for  i (1 : m)

  C[i] = X[i]
```

(Q3)

```
Calculate_Rotations (A) {
    start = 0;
    last = len(A) - 1

while ( start < end)
{
  mid = (start + last) / 2
 if ( mid[A] < A(end])
      return (mid + 1)
 else if
      {
        (A (mid ] < A(end))
        [ end = mid;
            }

 else
      { start = mid + 1
      }
return 0;
```

(Qu)

a)

```
is major(A,m){
    count = 0;
    for(i=0 to len(A))
        if(A[i]==m)
            count++
                }
    else if( count > len(A)/2)
        return count;

    else
        return 0
```

majority using divide and conquer

```
majority(A,p,q)
{   if(p=q)
        return p
    mid = p+q/2
    m1 = majority(A,p,mid)
    m2 = majority(A,mid+1,q)
    if(major(A,m2))
        return m2

    else
        return false
```

(Q4)

b)

```
find_majority (A1, A2, A)
    if A1 == A2
        return A1

    count A_1 = A.count(A1)
    count A_2 = A.count(A2)

    if count A_1 > count A_2
        return A1
    else
        return A2
```

(Q5)

	Heap Sort	merge sort	Quicksort
1	3909572	1968423	14932642
2	78423120	36386240	26579269
3	11843842	59041283	53420408
4	16243980	75629480	63436201
5	14289399	93411829	73421934

Average Heap sort = 50623974.2

Average merge sort = 50830591

Average Quick Sort = 46358090.8

(b)

Time Taken

Heap Sort	Merge Sort	Quick Sort
1860	969.842	289.019
700.612	1039.86	592.792
623.89	618.604	296.941
938.26	710.868	682.643
657.92	710.812	439.623

Heap Sort = 930.448 ms

Merge Sort = 813.96 ms

Quick Sort = 488.832 ms

Hence we can see that

Heap > Merge > Quick Sort.