Use rand()%100 to store random values to an array of n

```
int* randomArray(int n)
{
   int *A = new int [n];
   for(int i=0; i<n;i++)
   {
      A[i] = rand()%100;
   }
   return A;
}
```

```
int * A = randomArray(20);

displayArray(A, 20);
```

Implementation of Merge Sort

```
void merge(int a[], int leftBottom, int
leftTop, int rightBottom, int rightTop)
{
   int length = rightTop-leftBottom+1;
   int temp[length];
   int left = leftBottom;
   int right = rightBottom;

   for (int i = 0; i < length; ++i) {
      if (left > leftTop)
         temp[i] = a[right++];
      else if (right > rightTop)
         temp[i] = a[left++];
      else if (a[left] <= a[right])
         temp[i] = a[left++];
      else
         temp[i] = a[right++];
   }
   for (int i=0; i< length; ++i)
      a[leftBottom++] = temp[i];
}

/**********************/
void merge_sort(int a[], int left, int right)
{
 if(left >= right) return;
 else {
  int mid = (left + right)/2;
  merge_sort(a, left, mid);
  merge_sort(a, mid+1, right);
  merge(a, left,mid,mid+1,right);
   }
}
```

```
merge_sort(A,0,19);
displayArray(A, 20);
```

Implementation of Quick Sort

```
void swapArray(int* A, int i, int j)
{
    int temp;
    temp = A[i];
    A[i] = A[j];
    A[j] = temp;
}

/**********************/

int partition(int *A,int StartIndx,int endIndx)
{for(int i=StartIndx;i<endIndx;i++)
   {
         if(A[i]<=A[endIndx])
      {
         swapArray(A,i,StartIndx);
         StartIndx++;
      }
         }
   swapArray(A,StartIndx,endIndx);
   return StartIndx;
}

/**********************/
void quicksort(int *A,int Left,int Right)
{         if(Left<Right)
         {   int pivot= partition(A,Left,Right);
      quicksort(A,Left,pivot-1);
      quicksort(A,pivot+1,Right);
         }
}
```

```
quicksort(A,0,19);
   displayArray(A, 20);
```