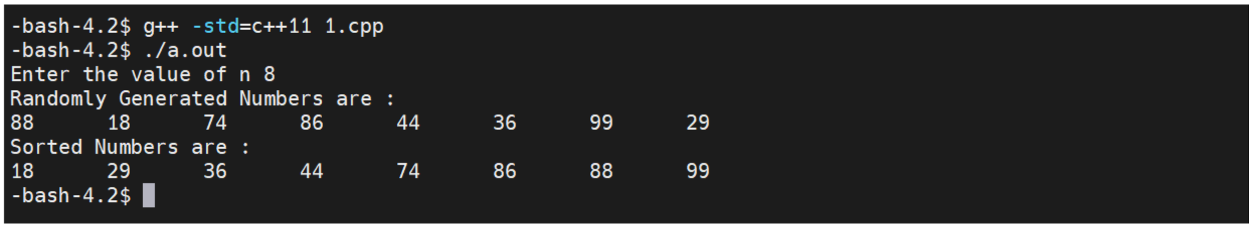
EECE7205 Fundamentals of Computer Engineering
HW1
Yuchao Su

P1

```
#include<iostream>
#include<random>
using namespace std;


void merge(int arr[], int p, int q, int r) {
    //If max in left is less than min in right, return
    if (arr[q] <= arr[q + 1])
        return;
    //left size
    int n1 = q - p + 1;
    //right size
    int n2 = r - q;
    //Allocate memory space
    int* L = new int[n1 + 1];
    int* R = new int[n2 + 1];
    //Copy to left
    for (int i = 0; i < n1; i++) {
        L[i] = arr[i + p];
    }
    //Copy to right
    for (int i = 0; i < n2; i++) {
        R[i] = arr[i + q + 1];
    }
    L[n1] = 2147483647;//infinity
    R[n2] = 2147483647;//infinity
    int i = 0;
    int j = 0;

    //Merge
    for (int k = p; k <= r; k++) {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
    }
//release memory space
    delete[] L;
    delete[] R;
}
```

```
void mergeSort(int arr[], int p, int r) {
    //Boundary
    if (p >= r)
        return;
    //Recursion
    mergeSort(arr, p, (p + r) / 2);
    mergeSort(arr, (p + r) / 2 + 1, r);
    merge(arr, p, (p + r) / 2, r);
}


int main() {
    random_device rd;  //Will be used to obtain a seed for the random number
engine
    mt19937 gen(rd()); //Standard mersenne_twister_engine seeded with rd()
    uniform_int_distribution<> dis(1, 100);//uniform distribution between 1
and 100
    int n;
    cout << "Enter the value of n "; // Entering Value of n
    cin >> n;

    int* A = new int[n];

    cout << "Randomly Generated Numbers are : " << '\n'; // Printing Randomly
Generated Numbers
    for (int i = 0; i < n; i++) {

        A[i] = dis(gen);
        cout << A[i] << '\t';
    }
    cout << "\n";
    cout << "Sorted Numbers are : " << '\n'; // Printing Sorted Numbers
    mergeSort(A, 0, n - 1);
    for (int i = 0; i < n; i++)
        cout << A[i] << "\t";
    cout << endl;
    return 0;
}
```

Result



```
-bash-4.2$ g++ -std=c++11 1.cpp
-bash-4.2$ ./a.out
Enter the value of n 8
Randomly Generated Numbers are :
88      18      74      86      44      36      99      29
Sorted Numbers are :
18      29      36      44      74      86      88      99
-bash-4.2$
```

P2

Pseudo code:

```
MergeSort(A,l,m,r)
    index=l
    i=0,j=0
    Let L[1..$n_1$+1] and R[1..$n_2$+1] be new arrays
    while i<=m-l+1 and j<=r-m
        if L[i]<R[j]
            A[index]=L[i]
            i++
        else
            A[index]=R[i]
            j++
        index++
    while i<=m-l+1
        A[index]=L[i]
        Index++
        i++
    while j<=r-m
        A[index]=R[j]
        Index++
        j++
```

P3

a. The two functions are both calculating the power of 2.
b. F2 is faster.
c. N and logn
   F1 need to calculate every number during recursion, while F2 have a memory to store the result.

P4

a. The purpose of this function is to sort the array in ascending order. The inner for loop moves the smallest element in the unsorted array to the front. The outer for loop removes the smallest element from the unsorted array. When the outer for loops n-1 times, the element was left is the biggest element in the array.
b. The worst case is the array is in descending order. So, we need to exchange the last one to front every time.

Step	Cost	Times
1	C1	1
2	C2	n-1
3	C3	n(n-1)/2
4	C4	n(n-1)/2

Total = C1+C2(n-1)+C3(n(n-1)/2)+C4(n(n-1)/2)
$O(n^2)$

P5

Pseudo code:

```
RecurInsert(A, n)
    If(n > 0)
        RecurInsert(A, n-1)
        Insert (n)
    else
        return
```

T(n)=T(n-1)+T(Insert)

Total = $\sum_{1}^{n}(T(n-1)+T(Insert))$

T(Insert)=n

Total = n^2