# 

# DSA Exercise Simple Sort Part 1

# SECJ2013 -02

# 

# Group 6 :

LEE SZE YUAN (A19EC0068)

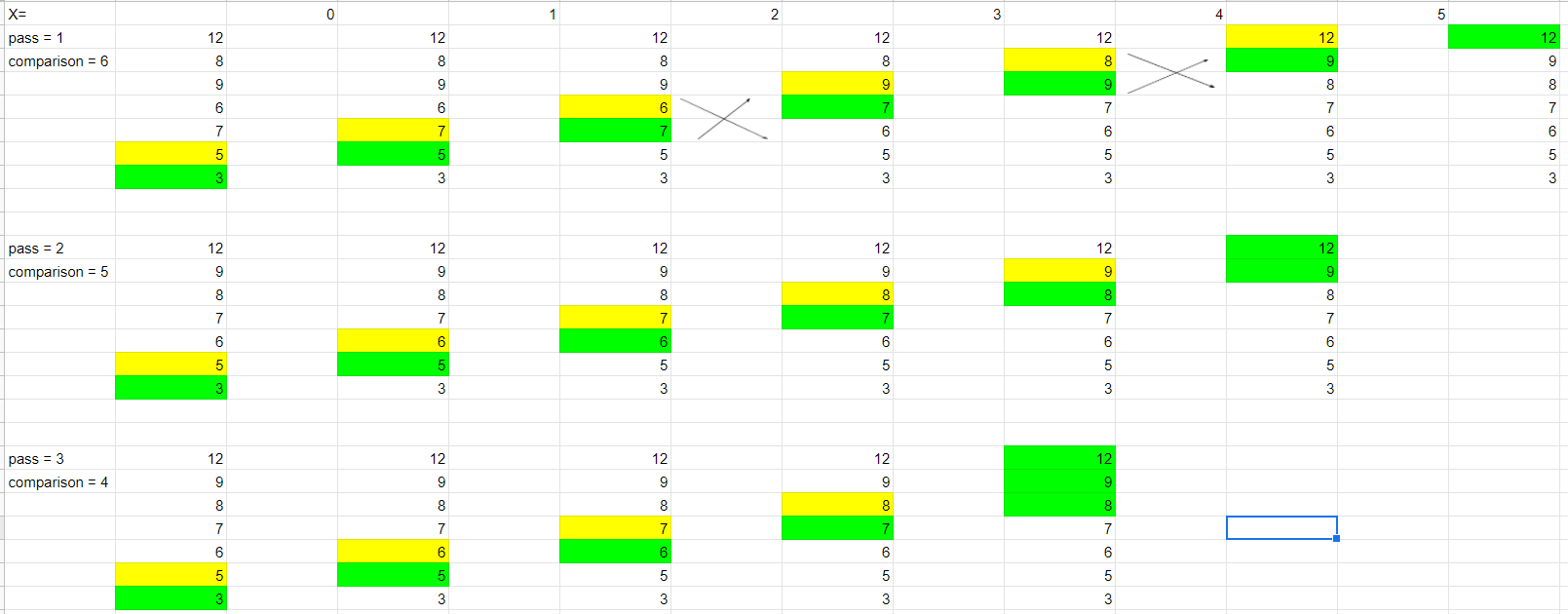
LOH YEW CHONG (A19EC0076)

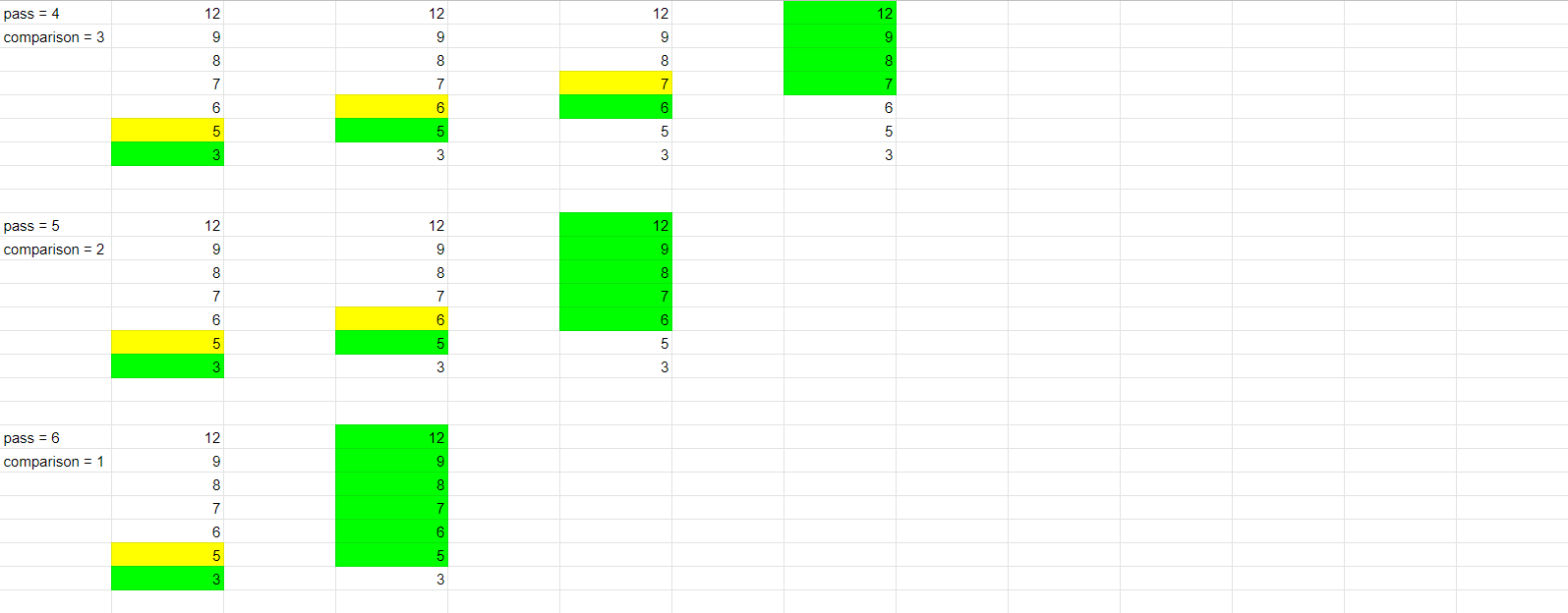
CHIA WEI HONG (A19EC0033)

# Exercise 1

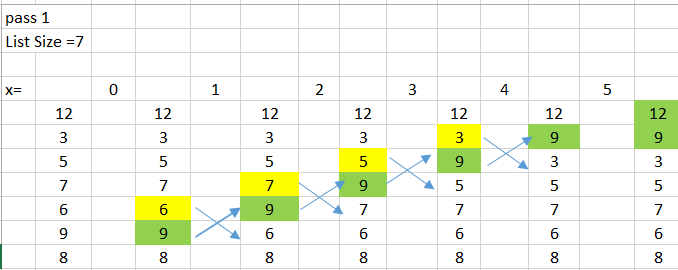
A.

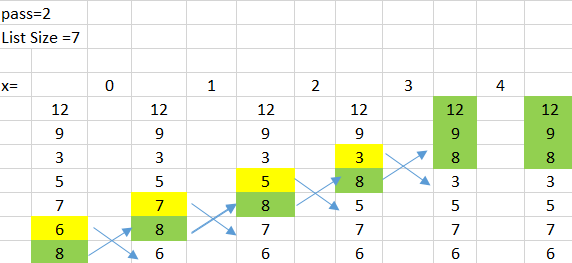
Array A

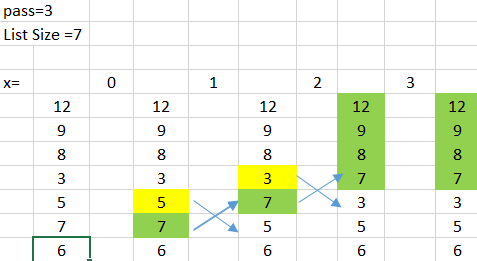


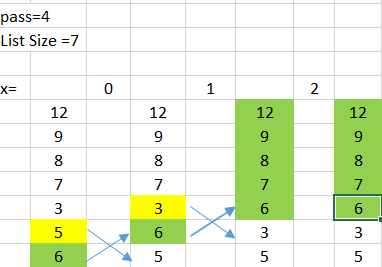


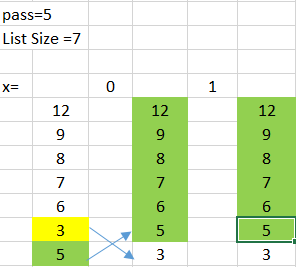
Array B

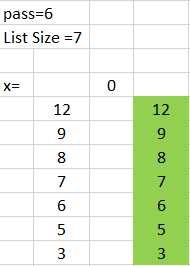












B.

Due to the design of algorithm, even after the array is sorted completely in the first few pass, the algorithm will still continue loop through the array until all the pass are completed

This can be improved by adding a conditional check in the algorithm. When there is no exchange of data inside the array(array is sorted completely) in certain pass, the condition is met, then the algorithm will stop looping through and exit the array

C.

#include <iostream>

using namespace std;

void BubbleSort(int data[], int listSize){

int tempValue;

bool is\_sorted = false;

for (int i = 1; i < listSize && !is\_sorted; i++){

is\_sorted = true;

for (int j=0; j<listSize-i; j++){

if (data[j] > data[j+1]){

tempValue = data[j+1];

data[j+1] = data[j];

data[j] = tempValue;

is\_sorted = false;

}

}

}

}

int main(){

int data[7] = {3,5,7,6,9,8,12};

BubbleSort(data, 7);

for (int i=0; i<7; i++){

cout << data[i] << " ";

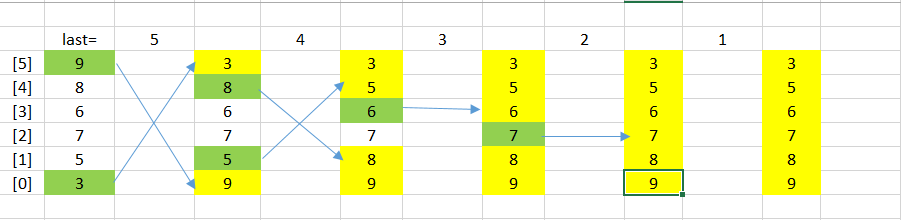
}

return 0;

}

# Exercise 3

A.



B.

Selection Sort

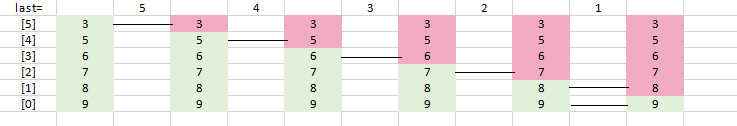
C.

Descending

D.

It chooses the largest or smallest item in the array and places the item in its correct place. Then, it chooses the next largest/next smallest time in the array and places the item in its correct place. This process will be repeated until all items are sorted.

E.



F.

The number of comparisons of both Array F and Array G is the same but the number of swapping is different . The number of swapping for Array F is higher compared to Array G which has no swapping.

Conclusion: Both of these cases have same time complexity of O(n2).