Team : Fantastic four

Deepti Yadawad	V00838513
James Beasley	V00803038
Chris Pouliot	V00830399
Devroop Banerjee	V00837868

Task 1 – Sorting

1. *How bubble sort works in my words.*

According to [1], bubble sort is a progressive comparison program which keeps on going until it concludes it is finished sorting. To break it down a bit more lets have a look at the following table:

Table 01;

1	***2***	6	4	3	7

As if bubble sort was tasked with sorting the above array it would start at the from left and then do a check to see if the secondary value, (the 2), to it's right is less then the primary selected value, (the 1).

Table 02:

1	***2***	***6***	4	3	7

The first check concluded that secondary number was greater and thus no swap was needed. Now the bubble moves all its checks one to the right and repeats, which once again results in no numbers being moved.

Table 03:

1	2	***6***	***4***	3	7
1	2	***4***	***6***	3	7

Now things are starting to get interesting as this check will result in it finding that the secondary value, (4), is less than the primary value, (6), is it saves the secondary value into a temporary variable then over writes the secondary with the primary value. Once that is done then primary is written over with the temporary variable.

Table 04:

1	2	4	6	3	***7***
1	***2***	4	3	6	7

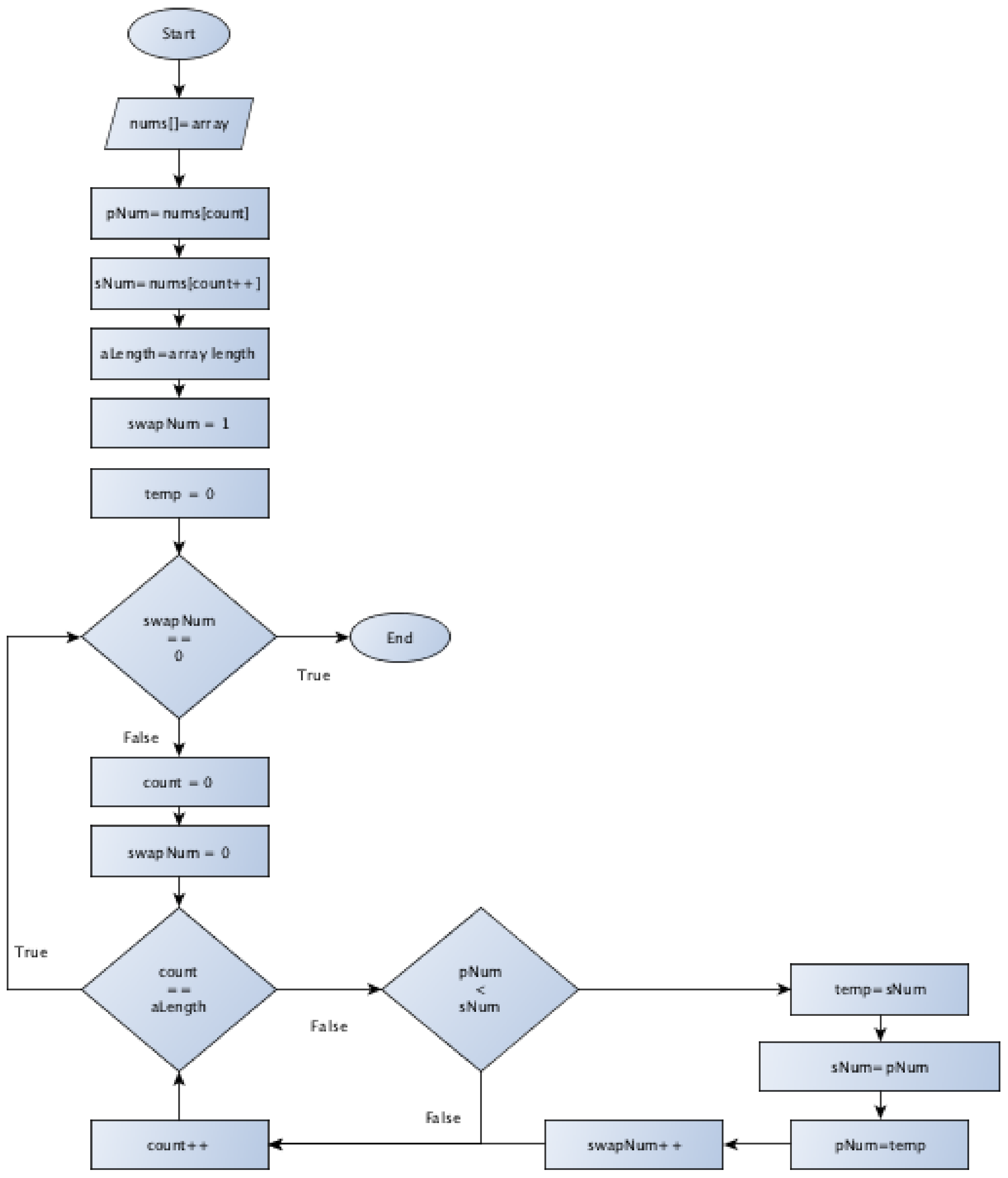
Once the end of the array is reached bubble simply starts over from the beginning. This process will continue until it manages to go through the entire array and not have to swap any values.

[1] = http://en.wikipedia.org/wiki/Bubble_sort

2. Pseudo Code for my bubble sort.

```
Start
        pNum=nums[count]
```

```
    sNum=nums[count++]
    aLenght=array length
    temp = 0
    swapNum = 1
    while (swapNum != 0)
        swapNum = 0
        count = 0
        while (count != aLength)
            if (pNum < sNum)
                temp = sNum
                sNum = pNum
                pNum = temp
                swapNum ++
            count++
End
```



3. ***Consider the following values using my bubble sort.***
   a. 5,4,3,2,1 would take 10 comparisons and 10 swaps
   b. 1,2,3,4,5 would take 10 comparisons and 0 swaps
   c. 10,3,8,2,5 would take 10 comparisons and 7 swaps

Task 2 - Picking the right sort of sort

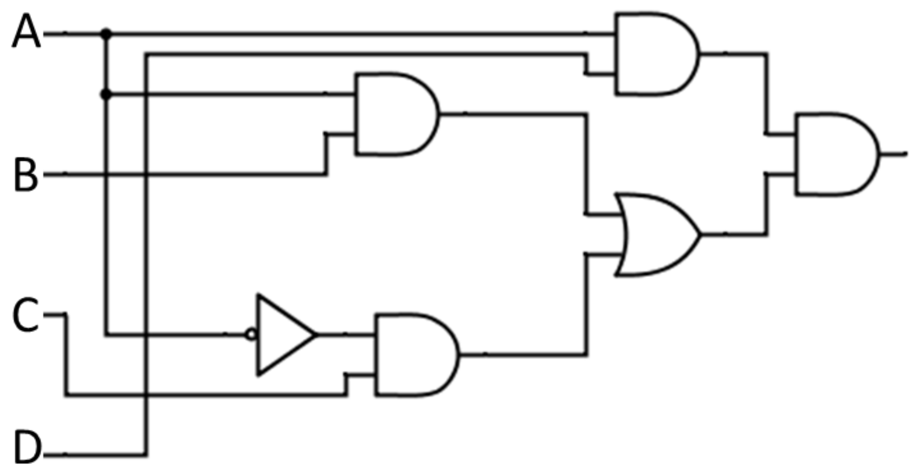
Characteristics	**Quick Sort**	**Insertion Sort**

Expending and replacing a character’s current power depending on the quest. Power associate with object transfers to character.	Selects a random pivot and sorts a set of data into order by swapping values according. It will take a lot of time and since there are only 2 indexes, this sorting method would be unnecessary.	It is a simple implementation especially since there are very few indexes (quest power and character power). It’s online so it will change the power value as soon as it receives it. It’ll also require a lot less place.
Power allows characters to move about the fantasy realm, create structures and tools, overcome other characters; helps the character solve the current quest.	Quick sort can help in this case, maybe by arranging the objectives in the quest according to the current power of the character. Quick sort is better with handling a larger number of indexes.	Insertion sort might not be able to help so much with this area of the game.
Every time a character gathers a new object a sorted list of that character's power objects is presented on the game screen, with the new object correctly inserted.	Quick Sort will be able to help sort the list of objects in ascending order according to the power of the object.	This method would be able to sort the objects correctly according to its power, and unlike quick sort, it would also be able to correctly insert any new objects into the list which has just been collected.
Any time a character expends power and uses up the power of an object, or picks up a new power object the list is sorted and presented again. As well, a sorted list can be requested by its owner any time throughout the game.	Quick Sort can help by sorting out all the objects when an object is picked up or when the list is called by the player.	Insertion sort can take comparatively long to sort out objects everytime the player calls the list, however it will always be up to date with the power values of the objects.

Overall, I believe Insertion Sort would help create a more efficient design, due to its ability to handle smaller sets of data well, its simplicity, adaptivity, stability, and low memory space requirement.

Task 3 - Logic Gate Design

1. **Diagram of Circuit**



2. **Truth Table**

A	B	C	D	A∧B	C∧¬A	(A∧B)∨(C∧¬A)	(A∧D)	((A∧B)∨(C∧¬A)) ∧(A∧D)
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	0	1	1	0	0
0	0	1	1	0	1	1	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0
0	1	1	0	0	1	1	0	0
0	1	1	1	0	1	1	0	0
1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0
1	0	1	0	0	0	0	0	0
1	0	1	1	0	0	0	1	0
1	1	0	0	1	0	1	0	0
1	1	0	1	1	0	1	1	1
1	1	1	0	1	0	1	0	0
1	1	1	1	1	0	1	1	1

Task 4 – Scratch (Basic Animation)

Project Notes: The Cat chases the bat around and everytime the cat pokes the bat, the bat says "Wow". Submitted as Task4.sb

Task 5 – Improving the Bubbler

The improved Bubbler introduces a new character who allows you to redeem yourself incase you get the incorrect answer. It allows you to play a game where the new character is supposed to catch the bat. The character can be moved around using WASD. If the character catches the bat, the bat says "NOOO".