ECE 368 Project 1
Name: Yi En Gan (Andrew)
Purdue login: gan35
Instructor: Felix Lin

Requirement:
gcc version 4.7 or higher
C11 standard
Support for new C functions such as getline(), itoa() and atoi()

```
gcc -std=c11 -o sorting sorting.c
```

On the latest version of gcc, the C11 standard is the default standard.

Introduction:
The program utilizes 'Shell Insertion' and 'Improved Bubble' to sort an unsorted list of integers from an input text file. The 'Shell Insertion' sort makes use of the three smooth number sequence generated using the formula pow(2, i) * pow(3, j) as the gap, while the 'Improved Bubble' sort uses a sequence of integers generated by dividing the previous integer by 1.3 as the gap.

Several improvements were made, such as the early termination of a sub shell sort when no swapping occurs in the first comparison. This is illustrated below.

Gap = 3
3 4 6 8 2 7 9 1 5
3 4 6 8 2 7 9 1 5 – increment comparison pair by one
3 2 6 8 4 7 9 1 5 – swapping occurs
3 2 6 8 4 7 9 1 5 – increment comparison pair by one
3 2 6 8 4 7 9 1 5 – increment comparison pair by one
3 2 6 8 4 7 9 1 5 – normally the next comparison would be 3 and, but given that no swapping occurred for '8' and '9', there is no point in comparing '3' and '8' since the same comparison has been done in step one and no values in the relevant positions were changed since.

The improved bubble sort utilizes the generated N / 1.3 sequence which significantly reduces the number of comparisons required as the elements can now move through multiple positions in the array at once instead of moving one position at a time using the gap=1 bubble sort.

Four header files are included, namely <stdio.h>, <stdlib.h>, <string.h> and <stdbool.h>.
Naming conventions such as INPUT_FILE, OUTPUT_FILE and SEQ_FILE are defined at the top.

The program consists of 6 functions and 2 helper functions. The Load_File and Save_File functions are called by several other functions in the program. The sequence generating functions generate and save the sequence as a list into a text file. Due to the difficulty in generating a sorted three smooth number sequence, the sequence is generated unsorted, and then Load_File and qsort are called to sort the sequence. As such, the shell sort starts reading its gap sequence from the end, while the bubble sort reads its gap sequence as usual.

Save_Seq1 Time Complexity
2 ‘for loops’ that map (x from 1 to N) to (y from 1 to N)
x and y may be referred to as the power of 2 and 3 respectively.
$O_1(n) = n^2$
The unsorted sequence is later sorted using qsort
$O_2(n) = \log n$
Therefore, the overall $O(n) = n^2 + \log n$

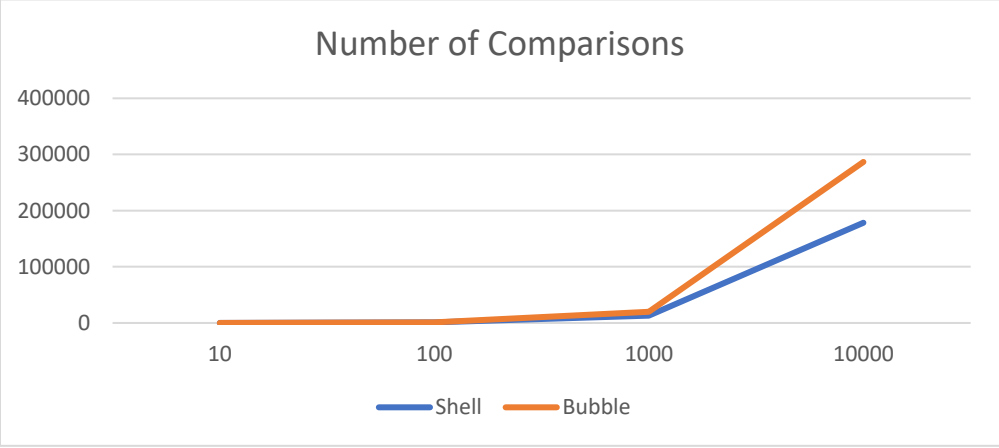
Save_Seq2 Time Complexity
A single ‘do while loop’ that takes unsorted array length and divides it by 1.3 until it reaches 1
$O(n) = \log n / \log 1.3 = \log n$

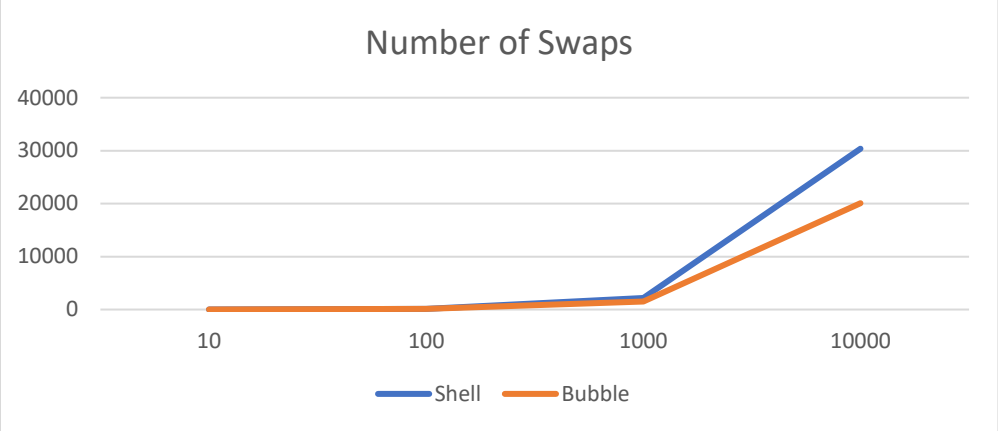
Save_Seq1 and Save_Seq2 Space Complexity
Consists of two malloc’d data types, both being char*
newChar is the newly added string while currChar is the overall string to be printed to file.
newChar is allocated length of unsorted array, as this is the largest possible value for the sequence.
currChar is allocated product of array length and single char size, as the sequence length can never exceed array length.
Maximum space allocation = n + n = 2n

Time Complexity of Sorting Algorithms
All were tested using worst-case scenario inputs (descending order)

	Comparison Count		Swapping Count	
Sample Size	Shell	Bubble	Shell	Bubble
10	33	32	7	7
100	809	1096	148	116
1000	13006	19706	2190	1536
10000	178453	286735	30368	20072

Except for cases where the sample size is extremely small, shell sort consistently required less comparisons to sort an array than bubble sort, while the bubble sort performed less swapping of two integers in its sorting process compared to the shell sort. Both algorithms showed exponential growth in time consumption as the sample size increased.





Space Complexity of Sorting Algorithms

While no malloc() was called in both sorting subroutines, the reading of the list of unsorted integers using Load_File() did require allocation of memory.

The allocated size of the list array is determined by the integer in the first line of the input text file, i.e. the number of integers in the unsorted list.

The long* list array is given memory allocation using the following line of code

```
long *list = malloc(sizeof(*list) * (*Size));
```

Space complexity = (4 * Size) bytes