**CSE 314**

**Online (A1)**

**Time: 50 minutes**

Consider a scenario where a buffer of integer is shared by 3 individual processes named as **Inserter**, **Sorter** and **Getter**. The buffer can be expandable, i.e., it can be a dynamic array with initial size, *n* = 0. The processes work as follows:

1. **Inserter** process inserts *n* (≥ 0) number of new items to the buffer, prints the buffer whatever it contains and sends a message (i.e., “Sort”) to **Sorter** process.
2. **Sorter** process receives the message from **Inserter** process, sorts the items of the buffer (ascending order) and sends a message (i.e., “Sorted”) to **Inserter** process. If the buffer is already sorted, then it sends a terminating message (i.e., “Terminate”) to **Inserter** process and terminates.
3. If **Inserter** process receives the message “Sorted” from **Sorter** process, then it appends *n* (≥ 0) new items to the buffer, prints the buffer and again get back to **Sorter** process. If it receives “Terminate” message from **Sorter** process, then it prints the items of buffer and terminates.
4. **Getter** process prints the first item of the buffer, if it is not empty. When **Sorter** process sorts the items of the buffer, it cannot access the buffer.
5. Each time take input (i.e., *n* and the *n* number of items) from the command prompt.

Write a program to implement the scenario where **Inserter, Sorter** and **Getter** are three child processes.

**N.B.:** Always print the status of the processes to trace their work. You can use any known sorting algorithm to sort the buffer. A sample code for sorting is given for your convenience.  
  
#include <bits/stdc++.h>

using namespace std;

int main()

{

    int arr[] = {1, 5, 8, 9, 6, 7, 3, 4, 2, 0};

    int n = sizeof(arr)/sizeof(arr[0]);

    sort(arr, arr+n);

}