CS 60
Programming Assignment #1
Due Date: April 8, 2010 (1:00 pm). 50 Points.
Remember: Points will be deducted for homework turned in after the Due Date.
Deadline: April 9, 2010 (1:00 pm)
Remember: No homework will be accepted after the Deadline
You must work independently on all the assignments.
TURN IN YOUR ASSIGNMENT USING THE TURNIN PROGRAM (SEE CLASS WEB PAGE FOR INSTRUCTIONS ON WHAT AND HOW TO TURNIN

Write a C program that has a value for n and n integers $a[0]$, $a[1]$, …, $a[n-1]$. The input and output format can be easily inferred from the sample input and output files for hw1 given in the map.html web page. Note that the input values are not necessarily sorted. Your program may assume that n is a positive integer whose value is between 1 and 1000. The integers $a[i]$ are between 1 and 100,000 for all $0 \leq i \leq n-1$.

Your C++ program initially is like the program given below that inputs n as well as the array $a[]$, and then sorts the array (see below the sort procedure). Then it must compute and print the following statistics.

- Compute and print the mode of the elements in $a[1..n]$ (the mode of a set of numbers $a[1..n]$ is one of the elements in $a[1..n]$ that occurs the most number of times). In case there is more than one mode, print the mode with the smallest value.
- Compute and print whether or not the mode is unique, i.e., tell us whether or not there are two or more elements in $a[1..n]$ that are repeated the most number of times.
- Count and print the number of elements with an odd value that appear only once in $a[0..n-1]$.
- Determine and print all the elements (in DESCENDING order) that are repeated an odd number of times in $a[0..n-1]$.
- Determine and print (in DESCENDING order) all the elements with an even value that appear exactly once in $a[0..n-1]$.
- Compute and print (in ASCENDING order) all the Fibonacci numbers that are elements of $a[0..n-1]$. The first Fibonacci number ($fib(1)$) is 0. The second Fibonacci number ($fib(2)$) is 1. The k^{th} Fibonacci number (for $k \geq 2$) is $fib(k) = fib(k-1) + fib(k-2)$. In other words, the first few Fibonnacci numbers are $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$.

For example if the numbers in the list are [2, 5, 6, 1, 5, 3, 6, 5, 6] then there are two modes (5 and 6). Your program should print a 5 for the mode. Of course the mode is not unique. The number of elements with an odd value that appear only once is one. The you will print elements 6, 5, 3, 2, and 1. The only element with an even value that appear exactly once is the element with value 2. The Fibonacci numbers in the array are: 1, 2 and 5. There are a couple of other examples in the map.html web page.

Below you will find an insert sort procedure. It also includes code to find the smallest and largest element of the (unsorted) array. This is code similar to the one you need to add. This code is available from the map.html web page and you may use it in your code. This version works correctly if you use the GNU C compiler, i.e, `gcc`

```
#include <stdio.h>

void Insert(int a[], int n, int x)
{// Insert x into the sorted array a[0:n-1].
   int i;
   for (i = n-1; i >= 0 && x < a[i]; i--)
      a[i+1] = a[i];
   a[i+1] = x;
}

void InsertionSort(int a[], int n)
{// Sort a[0:n-1].
   int i,t;
   for (i = 1; i < n; i++) {
      t = a[i];
      Insert(a, i, t);
      }
}

int main(void)
{  // defines the array with values
   int y[1000];
   int n, minval, maxval,i;
   scanf("%d",&n);
   printf("The number of elements is: %d\n",n);
   //read and print the elements y[0] ...
   for (i=0; i< n; i++)
     { scanf("%d",&y[i]);
       printf("%d\n",y[i]);
     }
   printf("\n");
   // computes the smallest and largest values in y[0] ...
   minval=y[0];
   maxval=y[0];
   for (i=0; i< n; i++)
      {if ( y[i] < minval ) minval = y[i];
       if ( y[i] > maxval ) maxval = y[i];
      };
   // prints the smallest value in y[0] ...
   printf("The smallest value is: %d\n",minval);
   // prints the largest value in y[0] ...
   printf("The largest value is: %d\n",maxval);
   InsertionSort(y,n);
   // prints the values in sorted order
   printf("The sorted numbers are: \n");
   for (i=0; i< n; i++)
      printf("%d\n",y[i]);
   printf("\n");
}
```

Turn in a hard copy of your program electronically together with a file named `student.id` that contains two lines (the first line has your name and the second one has your e-mail address).