Dylan Thornburg | ECEN 121 | HW 1

Problem 1:

Write a program that includes an array of 20 floating point numbers. Write 3 subroutines, one each to calculate the mean, the mode, and the median. Call these subroutines and print out the results.

Data set used: (0, 0.1, 0.2, …, 1.9)

CODE:

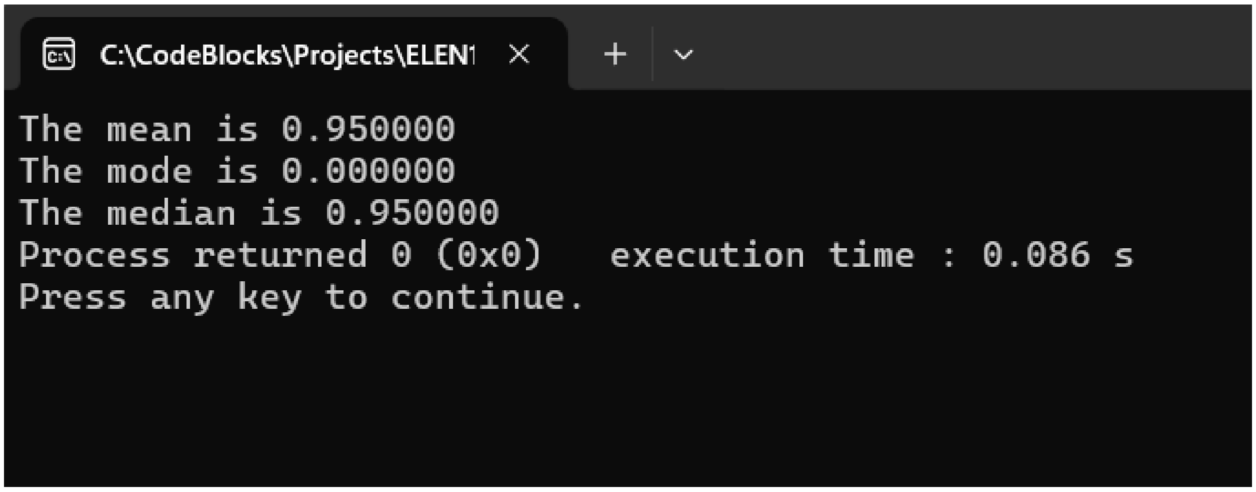
```
#include <stdio.h>
#include <stdlib.h>

float mean(float array[20])
{
   int i;
   float mean;
   for (i=0; i < 20; i++)
   {
      mean += array[i];
   }
   mean = mean/20;
   return mean;
}
float mode(float array[20])
{
   int max = 0;
   float mode = array[0]; //If all numbers in array show up only once, then the mode is just the first number in the
array
   int i;
   for (i = 0; i < 20; i++)
   {
      int count = 0;
      int j;
      for (j = 0; j < 20; j++)
      {
         if (array[j] == array[i])
            count++;
      }
      if (count > max) {
         max = count;
         mode = array[i];
      }
   }
   return mode;
}

float median(float array[20])
```

```
{
    float median = (array[9] + array[10])/2;
    return median;
}
void insertionSort(float array[20])
{
    int i;
    int j;
    float x;
    for (i = 1; i < 20; i++) {
        x = array[i];
        j = i - 1;
        while (j >= 0 && array[j] > x) {
            array[j + 1] = array[j];
            j = j - 1;
        }
        array[j + 1] = x;
    }
}
int main()
{
    // Simply initializing and filling up the array
    float array[20];
    float x = 0.1;
    int i;
    for (i=0; i < 20; i++)
    {
        array[i] = x * i;
    }
    //float array [20] = {0, 0.1, 0.7, 0.2, 1.2, 0.8, 0.4, 0.5, 0.6, 0.1, 0.3, 0.9, 1, 2, 1.2, 1.2, 1.8, 1.9, 1, 2}; used for next
data set
    insertionSort(array); //quick little insertion sort in case array wasnt sorted
    printf("The mean is %f \n", mean(array));
    printf("The mode is %f \n", mode(array));
    printf("The median is %f", median(array));
    // Simply initializing and filling up the array
}
```

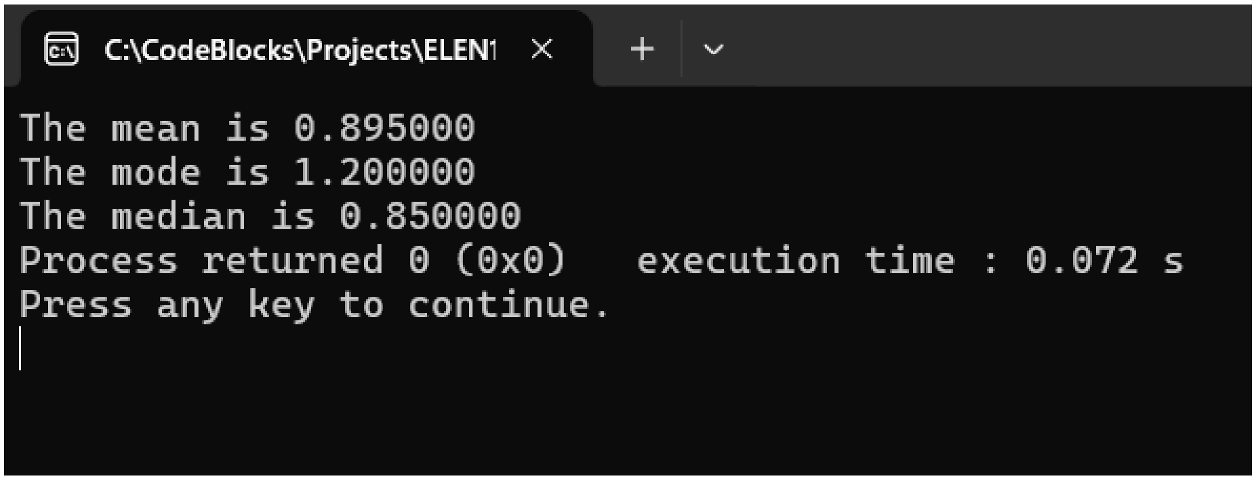
Results:



```
The mean is 0.950000
The mode is 0.000000
The median is 0.950000
Process returned 0 (0x0)   execution time : 0.086 s
Press any key to continue.
```

Mean check: 0 + 0.1 + 0.2 + … + 1.9 = 19; 19/20 = 0.95
Mode check: all numbers are only shown once, thus we just take the first number in the array which is 0.
Median check: array[9] = 0.9; array[10] = 1; 1+0.9=1.9; 1.9/2 = 0.95

Now I will use a different data set. One where numbers will actually be repeated
Set: 0, 0.1, 0.7, 0.2, 1.2, 0.8, 0.4, 0.5, 0.6, 0.1, 0.3, 0.9, 1, 2, 1.2, 1.2, 1.8, 1.9, 1, 2
Results:



```
The mean is 0.895000
The mode is 1.200000
The median is 0.850000
Process returned 0 (0x0)   execution time : 0.072 s
Press any key to continue.
```

Mean check: All numbers added together equal 17.9; 17.9/20 = 0.895
Mode check: 1.2 appears the most at three times
Median check: array[9] = 0.8 and array[10] = 0.9; (0.8+0.9)/2=0.85

Problem 2:
Write a subroutine that takes as its parameter an unsigned long integer representing the number of seconds since the beginning of 1950. Print out the month, day, year, and time. You can assume that days are exactly 24 hours. Test this subroutine for the following input values.

652,546,067
71,031,077
73,292,866
420,696,291
216,085,900
1,591,050,919
1,577,964,071
2,298,416,818

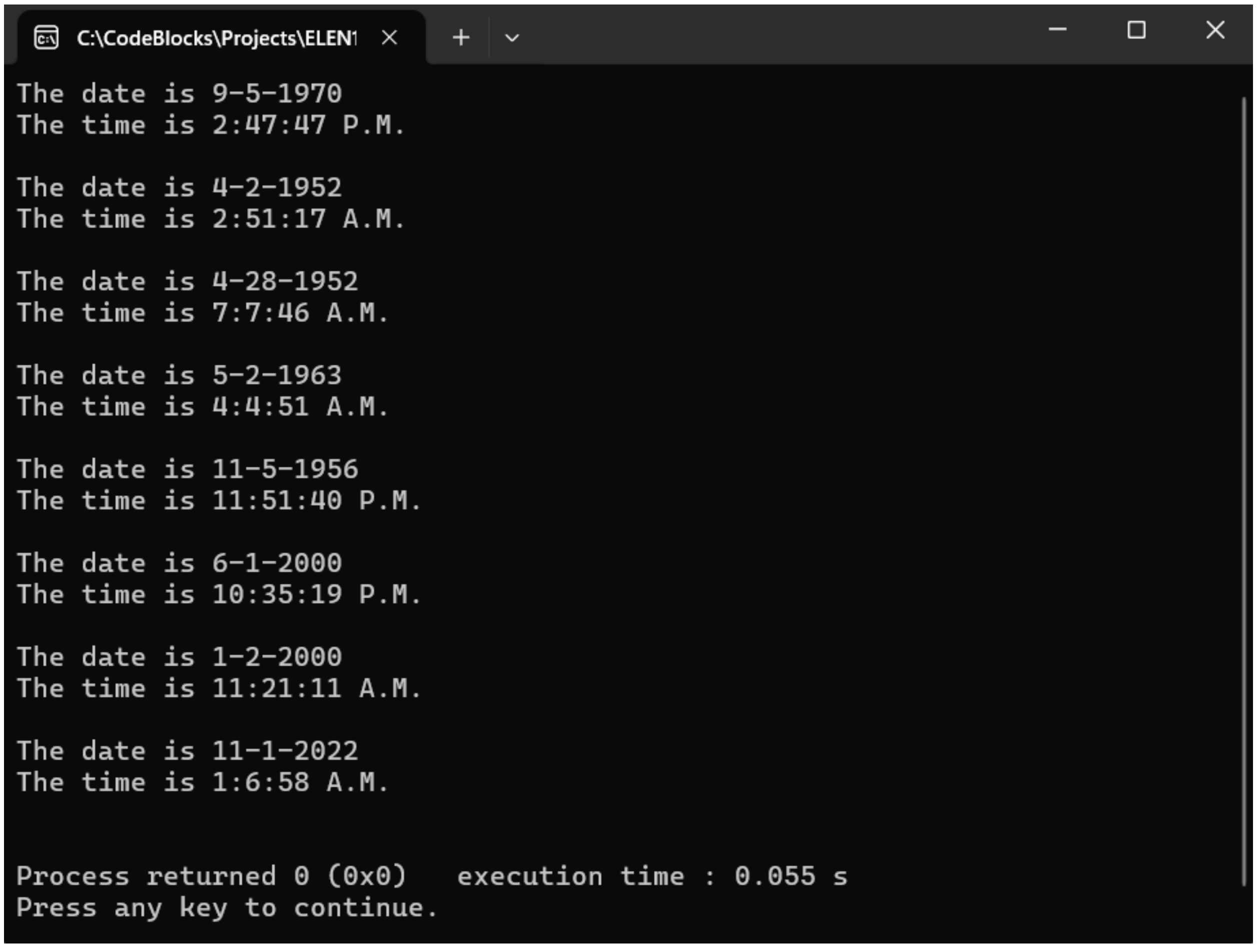
The above instructions say write a subroutine that takes an unsigned long integer as a parameter, however because we have 8 test cases, I made it take an array of unsigned long integers as its parameter to speed up the testing process.

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
void subroutine(unsigned long array[8])
{
    unsigned long newArray[8];
    int year_base = 1950;
    int year_const = 60*60*24*365;
    int leap_year_const = 60 *60*24*366;
    int four_year_const = leap_year_const + 3*year_const;
    int i;
    for (i=0; i < 8; i++)
    {
        int year;
        int month = 1;
        unsigned long temp = array[i];
        for(year=1950; ((temp >= year_const) && (year % 4 !=0)) || ((temp >= leap_year_const) && (year % 4 == 0));
year++) //year calc
        {
            if (year % 4 == 0)
            {
                temp = temp - leap_year_const;
            }
            else
            {
                temp = temp - year_const;
            }
        }
        int cond = 0;
        int feb;
        int month_31 = 31*24*3600;
        int month_30 = 30*24*3600;
        if (year % 4 ==0)
        {
            feb = 29;
        }
        else
        {
            feb = 28;
        }
        while(cond != 1) //month calc. a switch statement prolly would've been better in retrospect but this works and
would still take the same amount of time.
        {
            if(temp < month_31) // if in januray
            {
                cond = 1;
                continue;
            }
            temp = temp - month_31;
            month++;
            if (temp < feb*24*3600) // if in february
            {
                cond = 1;
                continue;
            }
            temp = temp - feb*24*3600;
            month++;
            if (temp < month_31) // if in march
            {
                cond = 1;
                continue;
            }
            temp = temp - month_31;
            month++;
            if (temp < month_30) // if in april
            {
                cond = 1;
                continue;
            }
            temp = temp - month_30;
            month++;
            if (temp < month_31) // if in may
            {
                cond = 1;
                continue;
            }
            temp = temp - month_31;
            month++;
            if (temp < month_30) // if in june
            {
                cond = 1;
                continue;
            }
            temp = temp - month_30;
            month++;
            if (temp < month_31) // if in july
            {
                cond = 1;
                continue;
            }
            temp = temp - month_31;
            month++;
            if (temp < month_31)// if in august
            {
                cond = 1;
                continue;
            }
            temp = temp - month_31;
            month++;
            if (temp < month_30) // if in september
            {
                cond = 1;
                continue;
            }
            temp = temp - month_30;
            month++;
            if (temp < month_31) //if in october
            {
                cond = 1;
                continue;
            }
            temp = temp - month_31;
            month++;
            if (temp < month_30) // if in november
            {
                cond = 1;
                continue;
            }
            temp = temp - month_30;
            month++;
            // else in december
            cond = 1;
        }
        int day = 1;
        while (temp > 24*3600) // day calc
        {
            day++;
            temp = temp - 24*3600;
        }
        int hour = 0;
```

```
        while (temp > 3600) // hour calc
        {
            hour++;
            temp = temp - 3600;
        }
        int min = 0;
        while (temp > 60) // min calc
        {
            min++;
            temp = temp - 60;
        }
        int seconds = temp; //remaining seconds are just the seconds (<60)
        printf("The date is %d-%d-%d \n", month, day, year); //formatting for printing
        if(hour < 12)
        {
            if (hour == 0)
            {
                hour = 12;
            }
            printf("The time is %d:%d:%d A.M.\n", hour, min, seconds);
        }
        else
        {
            hour = hour-12;
            if(hour ==0)
            {
                hour = 12;
            }
            printf("The time is %d:%d:%d P.M.\n", hour, min, seconds);
        }
        printf("   \n");
    }
}
int main()
{
    unsigned long array[8] =
    {
    652546067,
    71031077,
    73292866,
    420696291,
    216085900,
    1591050919,
    1577964071,
    2298416818
    };
    subroutine(array);
    return 0;
}
```

Results:



C:\CodeBlocks\Projects\ELEN1

```
The date is 9-5-1970
The time is 2:47:47 P.M.

The date is 4-2-1952
The time is 2:51:17 A.M.

The date is 4-28-1952
The time is 7:7:46 A.M.

The date is 5-2-1963
The time is 4:4:51 A.M.

The date is 11-5-1956
The time is 11:51:40 P.M.

The date is 6-1-2000
The time is 10:35:19 P.M.

The date is 1-2-2000
The time is 11:21:11 A.M.

The date is 11-1-2022
The time is 1:6:58 A.M.


Process returned 0 (0x0)   execution time : 0.055 s
Press any key to continue.
```

Problem 3:
Write a C program using the data structures below that will print out the strings in alphabetical order. Demonstrate that it works for both of my test sets.

Test set 1

char str1[] = "Hello";
char str2[] = "Dog";
char str3[] = "Cat";
char str4[] = "rabbit";
char str5[] = "man";
char str6[] = "woman";

char str7[] = "person";
char str8[] = "camera";
char str9[] = "TV";
char str10[] = "clueless";
char *slist[10] = {str1, str2, str3, str4, str5, str6, str7, str8, str9, str10};

Test set 2

char str1[] = "Andrew ";
char str2[] = "aardvark ";
char str3[] = "airplanes";
char str4[] = "America";
char str5[] = "air ball";
char str6[] = "Air Canada";
char str7[] = "airplane";
char str8[] = "Air Bud";
char str9[] = "apple";
char str10[] = "advantage";
char *slist[10] = {str1, str2, str3, str4, str5, str6, str7, str8, str9, str10};

CODE:

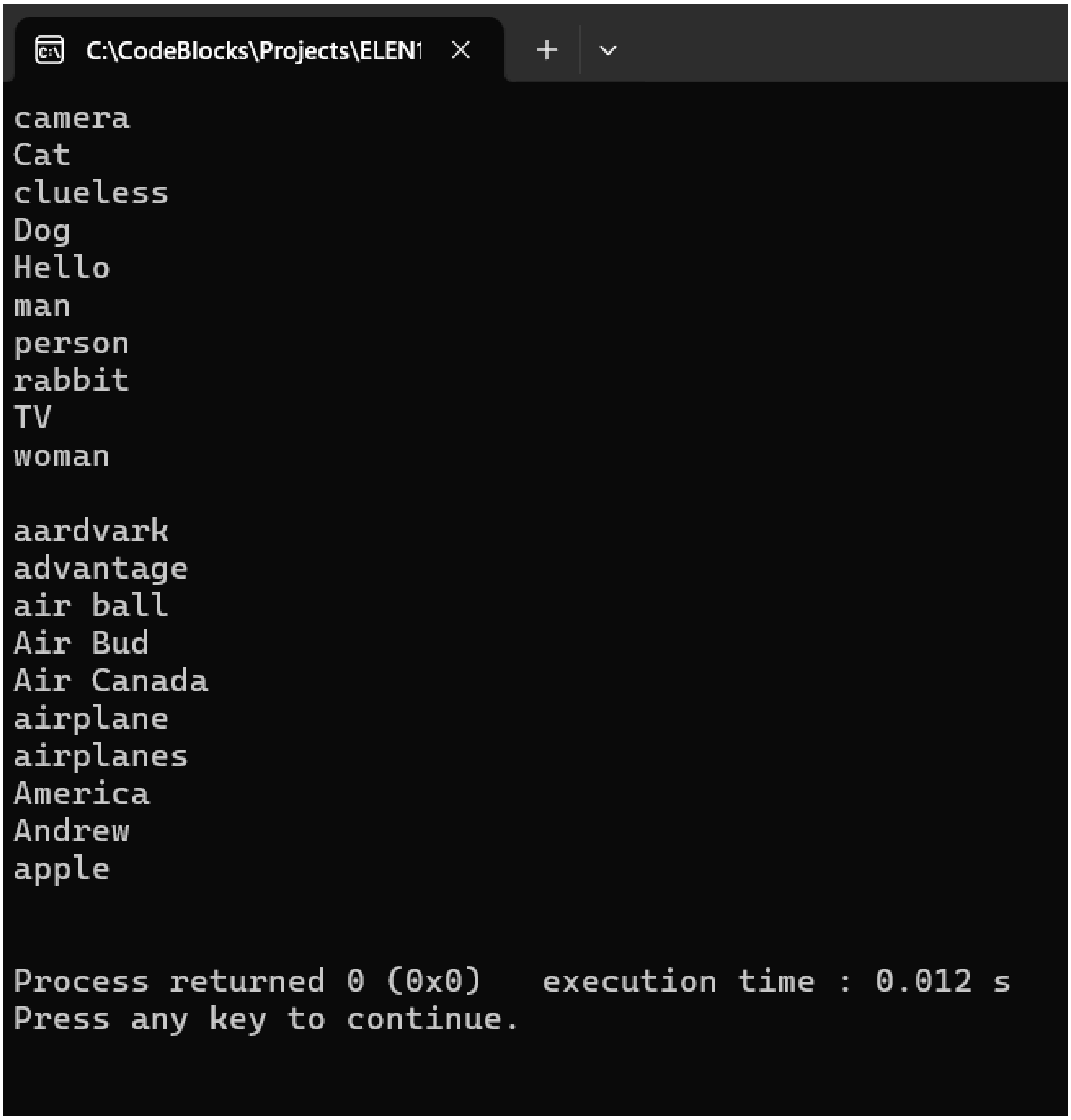
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void sorter(char**list)
{
  //doing this manually; screw qsort
  int i;
  int j;
  for (i = 0; i < 10; i++)
  {
    for(j=i; j < 10; j++)
    {
      if(strcasecmp(list[i], list[j]) > 0) //strcasecmp makes it so I dont have to adjust for cases; it simply ignores
case
      {
        char * temp = list[i];
        list[i] = list[j];
        list[j] = temp;
      }
    }
  }
  for (i = 0; i < 10; i++)
  {
    printf("%s \n", list[i]);
  }
  printf("\n");
}
int main()
{
  char str1[] = "Hello";
  char str2[] = "Dog";
  char str3[] = "Cat";
  char str4[] = "rabbit";
  char str5[] = "man";
  char str6[] = "woman";
  char str7[] = "person";
  char str8[] = "camera";
  char str9[] = "TV";
  char str10[] = "clueless";
  char *slist[10] = {str1, str2, str3, str4, str5, str6, str7, str8, str9, str10};

  sorter(slist);

  char str11[] = "Andrew ";
  char str12[] = "aardvark ";
  char str13[] = "airplanes";
  char str14[] = "America";
  char str15[] = "air ball";
  char str16[] = "Air Canada";
  char str17[] = "airplane";
  char str18[] = "Air Bud";
  char str19[] = "apple";
  char str20[] = "advantage";
  char *slist2[10] = {str11, str12, str13, str14, str15, str16, str17, str18, str19, str20};

  sorter(slist2);
}
```

Results:



```
camera
Cat
clueless
Dog
Hello
man
person
rabbit
TV
woman

aardvark
advantage
air ball
Air Bud
Air Canada
airplane
airplanes
America
Andrew
apple


Process returned 0 (0x0)   execution time : 0.012 s
Press any key to continue.
```