```
Script started on Sun 09 Apr 2017 11:18:12 PM CDT
\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ pwd
/home/students/b_pepa/CSC122/median
\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ cat median.info
Brandon Pepa
CSC122-001
Smack in the Middle of Nowhere
Lab

(level 1.5)

Description:
        The function of this program is to enter data from a file in
order to find the median of the data (must be sorted).

**Extra credit**
\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ cat median.cpp
#include <iostream>
#include <fstream>
using namespace std;

const long MAX_FNAME = 200;

//Bubble sorts data given a pointer and the size of the array
void sort(double * & p, size_t size);

//reallocates memory from the specified pointer p, to a new
//location of size new_size. will repoint p to new array.
bool reallocate(double * & p, size_t new_size);

//finds the median of a sorted set of data given an array and the size.
double get_median(double * & p, size_t size);

//returns the number of data values from a file. used to allocate memory
size_t length(ifstream & f);

//Puts the values of all the data into our array
void get_data(double * & p, ifstream & f);

//swaps 2 doubles (makes bubble sort code more steamline)
inline void swap(double & x, double & y)
{
   double t = x;
   x = y;
   y = t;
   return;
}



int main(void)
{
   //input stream for the file
   ifstream infile;

   //name of the file
   char fname[MAX_FNAME];

   //Pointer to array where the data will be stored
   double * pdata = NULL;

   //Represents how long the data is in the file
   size_t size;

   cout << "\tWelcome to the Median finding Program";

   //Opens and makes sure file is valid
   cout << "\n\nPlease enter the name of your data file: ";
   cin.getline(fname, MAX_FNAME);
   infile.open(fname);
   while(!infile)
   {
      infile.close();
      infile.clear();
      cout << "\nI'm sorry, I could not open \'" << fname << "\'."
              " Please enter another name: \n";
      cin.getline(fname, MAX_FNAME);
      infile.open(fname);
   }

   cout << "\nFile \'" << fname << "\' opened successfully!";

   size = length(infile);

   //allocates data to size + 1 so we can store the median at the end of the
   //data for convenience and efficiency.
   if(reallocate(pdata, size + 1))
   {
      get_data(pdata, infile);
      sort(pdata, size);

      *(pdata + size) = get_median(pdata, size);

      cout << "\n\nThe median of the data is " << pdata[size] << ".\n";
   }
   else
   {
      cout << "\n\aUnable to allocate space for " << size
           << " values!\n\n"
           << "Please shut down other applications first..." << endl;
   }

   delete [] pdata;
   pdata = NULL;
   infile.close();
return 0;
}



//Bubble sorts data given a pointer and the size of the array
void sort(double * & p, size_t size)
{
   bool done = false;
   size_t i = 0;
   while(i < size && !done)
   {
      done = true;
      for(size_t j = 0; j + i + 1 < size; ++j)
      {
         if(p[j] > p[j+1])
         {
            swap(p[j], p[j+1]);
            done = false;
         }
      }
      ++i;
```

```
   }
return;
}

//reallocates memory from the specified pointer p, to a new
//location of size new_size. will repoint p to new array.
bool reallocate(double * & p, size_t new_size)
{
   delete [] p;
   bool okay = false;
   p = new double [new_size];
   if (p != NULL)
   {
      okay = true;
   }
return okay;
}

//finds the median of a sorted set of data given an array and the size.
double get_median(double * & p, size_t size)
{
   if(size%2 == 1)
   {
      return p[size/2];
   }
   else
   {
      return (p[size/2 - 1] + p[size/2])/2;
   }
}

//returns the number of data values from a file. used to allocate memory
size_t length(ifstream & f)
{
   size_t l = 0;
   double t;
   f >> ws;
   while(!f.eof())
   {
      f >> t >> ws;
      l++;
   }

   //reset the file to the beginning so we can input data from this file
   f.clear();
   f.seekg(0);
return l;
}

//Puts the values of all the data into our array
void get_data(double * & p, ifstream & f)
{
   size_t i = 0;

   //Since this is the SAME way we checked for the length of the file,
   //we can assume it is safe and we wont run out of space in p
   f >> ws;
   while(!f.eof())
   {
      f >> *(p + i) >> ws;
      ++i;
   }

   //reset the file to the beginning incase we need to look at the data agian
   f.clear();
   f.seekg(0);
return;
}
```

```
\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ cat data1\033[K033[K033[K033[K
\033[K033[K033[K033[K033[K033[K\007\007\007\007\007CPP median
median.cpp***


\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ cat data1
1
2
3
4
5
6
7
8
9
\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ ./data\033[K033[K033[K033[K033[K
median.out
        Welcome to the Median finding Program

Please enter the name of your data file: bad.dat

I'm sorry, I could not open 'bob.dat'. Please enter another name:
data1

File 'data1' opened successfully!

The median of the data is 5.
\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ cat\033[K033[K033[K033[Kcat data
2
5
2
6
1
8
7
3
9
4
\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ ./median.out
        Welcome to the Median finding Program

Please enter the name of your data file: data2

File 'data2' opened successfully!

The median of the data is 5.
\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ cat data3
15
62
84
32.3
73
3
2
54
```

```
85
3
7
42
38
54
37
23
90
86
64
43
\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ vi\033[K033[K033[K\007./median.out
        Welcome to the Median finding Program

Please enter the name of your data file: data3

File 'data3' opened successfully!

The median of the data is 42.5.
\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ cat dà033[K033[Kmedian.tpq
Though Provoking Questions

1.What (data) types of values can your program handle?
    The program will be able to handle doubles or longs. It will use double
    as the main data type becuase the median could be between 2 numbers
    and so it wouldn't be gauranteed to be a long.

2.How do you count the number of data items which are in the file? Once
  you've counted them, how do you start over and read them in?
    The data could be counted with an eof loop and just read into a temporary
    variable and count how many times it was able to process. to start over
    you must seekg() the beginning of the file.

3.How do you allocate memory on the heap? Is it gauranteed to work?
    memory is allocated with the 'new' keyword. it isn't gaurenteed to work
    so if it fails (the pointer would be NULL) then the user must shut down
    other applications before the program can properly run with dynamic
    memory.

4.When do you allocate memory for your values array? Before or after counting
  values in the file?
    the memory should be allocated after counting the values in the file so
    you know how much memory needs to be allocated.

5.When you are done with dynamic memory, what should you do?
    the memory shoudl be deallocated after we're done using it so it doesn't
    clog system resources and can be used for other applications and memory
    allocation.

6.What's this NULL value for, anyway?
    It's a value that we can store in a pointer that represents that the
    memory is not properly allocated for this pointer (so we can't use it
    to store things) the system also gives the pointer a NULL value if it
    wasn't able to allocate memory

7.Which sorting algorithm did you use? How efficient is it? (i.e.What is its
  complexity?)
    I'm using bubble sort because of it's simplicity and the fact that we're
    not going to be using large amounts of data to find the median for. the
    complexity is the number of comparisons the program has to do, and for
    bubbe sort it's n^2.

8.Does it matter if the data are sorted in ascending or descending order?
  Why/Why not?
    It doesn't matter if the data is sorted in ascending or descending order
    because the middle of the data will always be the same number or 2
    numbers in the data. the middle is in the same place no matter what.

9.Do your median, sort, etc. functions need to know that your data array is
  dynamic?
    the median and sort only need to know the length of the data. None of the
    functions actually have to know that the array is dynamic because the
    data doesn't need to change lengthe when executing the function.
\033]0;b_pepa@mars:~/CSC122/median\007[b_pepa@mars median]$ exit
exit

Script done on Sun 09 Apr 2017 11:20:15 PM CDT
```