Ahmad Mohammad

c1001633 / asm6t

proj1

		deduction
TURNIN TIME	turned in on time.	0 %
SOURCE CODE SEARCH RESULTS	MISSING: None. FOUND: doubleArray, nums.txt, calcAvg	0

DEDUCTIONS	-0
FINAL GRADE	100

COMPILATION LOG

```
c1001633 MAKE PASS

MAKELOG OF PROVIDED MAKEFILE:
g++ -c -g -O0 -std=c++11 -Wall proj1.cpp -o proj1.o
g++  proj1.o -o proj1
```

```
c1001633 runlog begin:
Test 1:
Output:
200 301.075
400 381.645
800 513 416.214
```
✓

```
Test 2:
Output:
200 300.475
400 318 349.755
```
✓

```
Test 3:
Output:
200 361.985
400 373.022
```
✓

800 601 421.378

```
Manifest: proj1.cpp.lst
-----------------------
-----------------------
 Jan 24 21:56 proj1.cpp
```

```cpp
// Ahmad Mohammad
// CSCI 3110-001
// Project 1
// Due: 01/25/2022
// Desc: This program will dynamically allocate an array (heap) of 200 int
// and take numbers from each line of "nums.txt" and add them to the array,
// once the array reaches its capacity it will double until it reaches eof
// each iteration of the while loop it will write to "p1out.txt" the
// important data (size, amt read, avg)
#include<iostream>
#include<fstream>


using namespace std;

// Function Prototypes
double calcAvg(int * dynarr, int amtval);
int * doubleArray(int * oldarr, int * size);


int main()
{
               // Declaring files and setting them equal to files used
    ifstream infile;
    ofstream outfile;
    infile.open("nums.txt");
    outfile.open("p1out.txt");

               // declaring size, and dynamically allocating array (arr) of length size
    int size = 200;
    int *arr = new int[size];

               //declaring variables needed in loop.
    int line; // number in input file
    int i = 0;  // counter used for index of array
    double avg; // used to receive return from doubleArray()

               // Loop through each line in infile and copy to our array until
               // infile is done. Whenever we reach capacity of our array we will
               // cout and write to our file, the size and average of the
               // elements currently in our array.
               while(infile >> line)
    {
        arr[i] = line;
        i++;
        if(i == size)
        {
            avg = calcAvg(arr, i);
                                             outfile << size << " " << avg << endl;
            cout << size << " " << avg << endl;
            arr = doubleArray(arr, &size);
        }

    }

               // calculating final avg and cout final results and writing final
               // results to outfile
    avg = calcAvg(arr, i);
    cout << size << " " << i << " " << avg << endl;
    outfile << size << " " << i << " " << avg << endl;
```

```

               // deallocate memore for arr and set equal to NULL
               delete [] arr;
               arr = nullptr;

               // close files
               infile.close();
               outfile.close();

               return 0;
};



// This function will calculate the average of the dyn. array and takes
// the amount of values in the array, and the array itsself as parameters.
// To do this we loop through the array and add each index to a variable and
// divide by amt val.
double calcAvg(int * dynarr, int amtval)
{
               // return variable
    double calcavg;

               //loop to get sum of all array vals
    for(int i = 0; i < amtval; i++) // look at i if math goes wrong
        {
            calcavg += dynarr[i];
        }

               //getting average
    calcavg /= amtval;

    return calcavg;
}

// This functions purpose is to double the size of our old dynamic array
// by creating a new array double the size if the old one and copying each’
// index pos of the old to the new.
int * doubleArray(int *oldarr, int *size)
{
               // dereferencing size inorder to change the value of the pointer
    *size *= 2;

               // dynamically allocating new array with new (double) size.
    int *newarr = new int[*size];

               // for loop to copy values friom old to new
    for(int i = 0; i < *size; i++)
    {
        newarr[i] = oldarr[i];
    }

               //deallocating memory space for old array and setting that memeory = NU
LL
    delete [] oldarr;
    oldarr = nullptr;


    return newarr;
}
```