Taylor Coogan
ELE 408
Bin Li
February 14, 2017

Homework 1

Part 1:

```
/**
-Taylor Coogan
-rectangle.c
-This code takes the input of an integer and prints a
square of asterisks of that side length.
**/
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
   int ret = 0;   // for testing if sscan completed without error
   int side = 0; // variable for side length initialized to zero

   ret = sscanf(argv[1], "%d", &side); // scanning user input for integer value, stores in side
      if (ret != 1)return 0;                    // if this fails, exit program

   int i = 0;      // initialize loop counters
   int j = 0;
   printf("\n");   // print new line for formatting

         for(i = 1; i <= (side); i++)        // outer loop for rows
         {
                  for(j = 1; j <= (side); j++)        // inner loop prints columns
                  {
                           printf("* ");
                  }

                  printf("\n");      // after each row is printed, new line
         }

         getchar();       // just in case console wants to close
         return(0);       // exit program
}
```

Default Term + Browser

```
* * * * * *
* * * * * *
* * * * * *

sh-4.2$ gcc main.c -o p1
sh-4.2$ ./p1 8

* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
```

Part 2:

```
/**
-Taylor Coogan
-grade.c
-This code takes the input of a list of grades. It then puts them in an array,
sorts them to find min, max, and average, and then reports that info to console.
**/

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

// declaring functions used cause I think you need to do this in C
void getGrade(int argc, char *argv[]);
int findMaximum(int grade[]);
int findMinimum(int grade[]);
float calculateAverage(int grade[]);
void printResults(int maximum, int minimum, float average);

/* Now we declare global vars that will be needed */
int arraySize = 0;  // holds number of elements
int *grade = 0;     // pointer for array
int ret = 0; // for error checking when reading from argv

/* Our 3 main variables for results*/
int maximum = 0;    // int for maximum value
int minimum = 0;    // int for minimum value
float average = 0;    // float for average

int j = 0; // counter var global

int main(int argc, char *argv[])
{
        printf("\n"); // new line for formatting
    printf("Grades List: ");    // this part is for debugging
    getGrade(argc, argv);       // but I like it so i'm gonna leave it
    printf("\n");               // hopefully this is okay

    /* Now we call our important functions*/
    maximum = findMaximum(grade);
    minimum = findMinimum(grade);
    average = calculateAverage(grade);
```

```
    printResults(maximum, minimum, average);

            getchar();
            return(0);
}
/* Puts the grades into an array that is allocated in memory*/
void getGrade(int argc, char *argv[])
{
    arraySize = argc - 1;   // for setting array size to # of grades
    grade = (int*)calloc(arraySize, sizeof(int));   // allocate array in memory

    int i = 0;  // array counter
    for (i = 0; i < arraySize; i++) // loop through argv array to get grades
    {
        ret = sscanf(argv[i+1], "%d", &grade[i]); // get nums from argv[]
                                                  // put in grade array
        if (ret != 1) exit(0);                    // make sure no errors reading argv
    }

    for(i = 0; i < arraySize; i++) // I added this printing function
    {

        printf("%d ", grade[i]);    // for debugging

    }
}

/* Finds maximum value in array and returns it*/
int findMaximum(int grade[])
{
    int biggest = grade[0]; // set temp variable to first value

    for(j=0; j < arraySize; j++)    // loop through and keep largest
    {
        if(grade[j] >= biggest)
        {
            biggest = grade[j];
        }
    }

    return(biggest);    // return largest number
}
```

```
/* Finds minimum value in array and returns it*/
int findMinimum(int grade[])
{
   int smallest = grade[0];    // sets temp var to first value

   for(j=0; j < arraySize; j++)    // loops through all values
   {                               // comparing and keeping smallest
      if(grade[j] <= smallest)
      {
         smallest = grade[j];
      }
   }

   return(smallest);   // returns smallest value
}
/* Finds the average of all the grades*/
float calculateAverage(int grade[])
{
   float sum = 0;          // floating point var for running sum
   float floatSize = 0;    // for converting array size to float

   floatSize = (float)arraySize;   // we wanna do float / float i think

   for(j=0; j < arraySize; j++)    // loops through all values
   {                               // adding them to sum
         sum = sum + grade[j];
   }

   sum = sum / floatSize;  // divide sum by number of elements
   return(sum);        // return the average
}
/* Prints the results to the console*/
void printResults(int maximum, int minimum, float average)
{
   /* We print our results here*/
   printf("Maximum Grade: %d \n", maximum);
   printf("Minimum Grade: %d \n", minimum);
   printf("Average Grade: %.1f \n", average);

}
```

Default Term + Browser

```
sh-4.2$ gcc main.c -o p2
sh-4.2$ ./p2 1 2 3 4 5 6 7 8 9

Grades List: 1 2 3 4 5 6 7 8 9
Maximum Grade: 9
Minimum Grade: 1
Average Grade: 5.0

sh-4.2$ ./p2 90 87 56 74 32 12 77 98

Grades List: 90 87 56 74 32 12 77 98
Maximum Grade: 98
Minimum Grade: 12
Average Grade: 65.8
```

Part 3:

```
/**
-Taylor Coogan
-coinFlip.c
-This code takes no user input and flips 200 instances of a coin.
It then reports the total heads and fails to the user.  The number of coin
flips can be changed by modifying the flips variable.
**/

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

int flip();          // declaring the flip method called later


int main()
{
   int flips = 200; // number of coin flips
   int i = 0;  // for counter
   int result = 3; // result of coin flip
   int heads = 0; // total heads
   int tails = 0; // total tails
   srand(time(NULL));  // seeds random num generator with time
                       // so that it is "actually" random

   for( i = 0 ; i < flips ; i++ ) // for each flip
   {
      result = flip();        // we get the result of the flip
      if(result == 1) { heads++; }    // check if it is heads...
      else if( result == 0) {tails++; }   // or tails
   }

   /* Print results */
   printf("\n Total Heads: %d", heads);
   printf("\n Total Tails: %d \n", tails);

   return 0;
}

/* Our function for the actual coin flip */
```

```
int flip()
{
    return(rand() % 2); // random number mod 2, so it can only be a 0 or 1
}
```