COP2220
Project 4 – Random Number Statistics

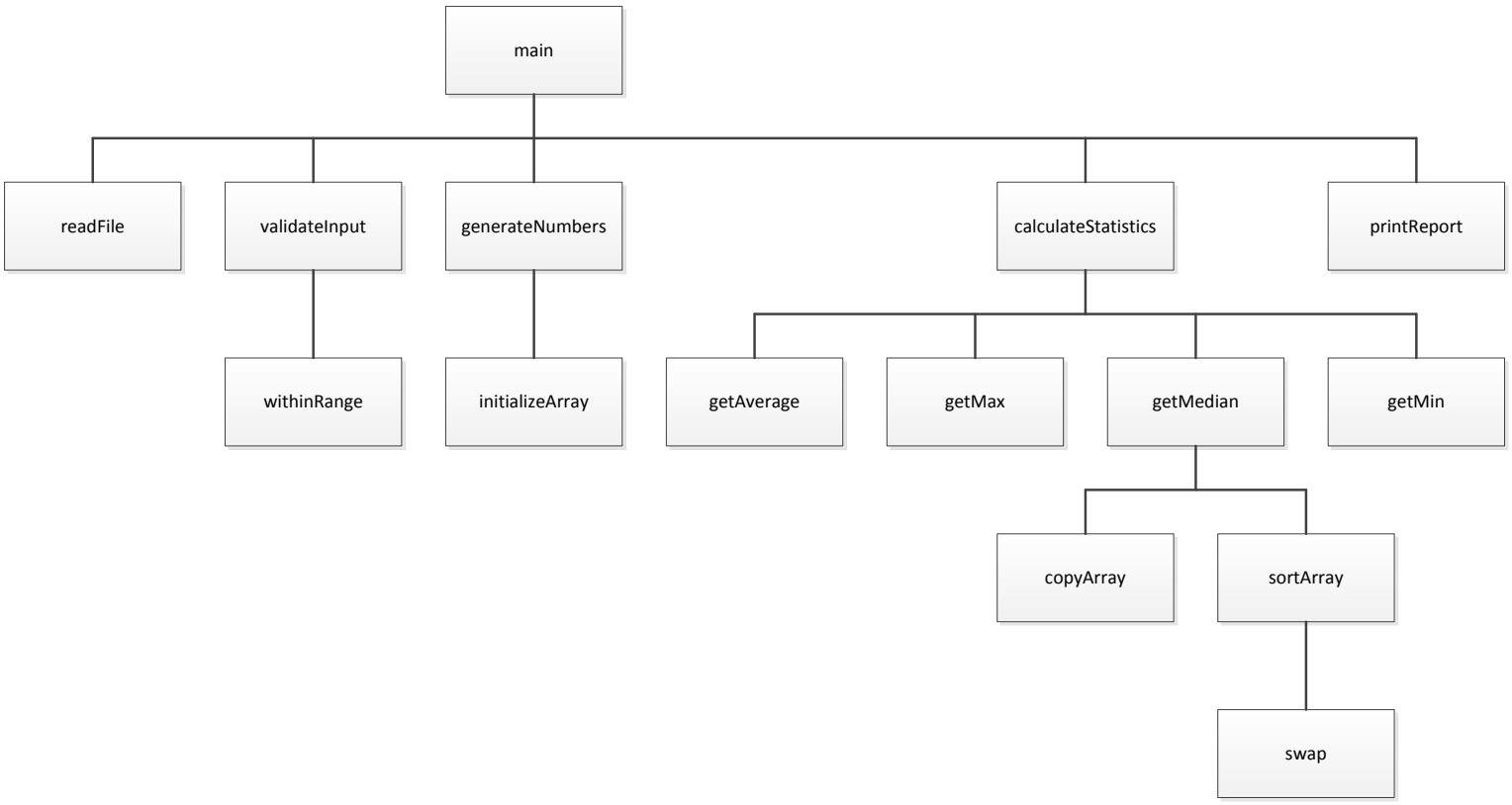
Submission Requirements

- Submit your project folder via the FileUploader utility provided on the course website
    - Follow the project submission guidelines for the class

Design Documentation Requirements

- Flowchart of the readFile function
- Pseudocode of the copyArray function

Design Specification Requirements




```
void calculateStatistics(int array[], int *pMin, int *pMax, double *pMedian, double *pAverage);
void copyArray(int array[], int copy[]);
void generateNumbers(int array[], int seed, int count);
double getAverage(int array[]);
int getMax(int array[]);
double getMedian(int array[]);
int getMin(int array[]);
void initializeArray(int array[]);
void printReport(int array[], int count, int min, int max, double median, double average);
bool readFile(char *filename, int *pSeed, int *pCount);
void sortArray(int array[]);
void swap(int array[], int pos1, int pos2);
bool validateInput(int seed, int count);
bool withinRange(int value, int minRange, int maxRange);
```

Note: Refer to the sample output in the Example Output section below.

1. The program must use the above 8 functions, as well as the main function
   A. **No additional functions are allowed**
2. The program must do the following things
   A. The program must accept 1 command line argument (the name of the input file)
   B. Read 2 strings from the file and convert them to their numeric equivalent (integer)
      1. First value is the random number generator seed
      2. Second value is the number of random numbers to generate
   C. Verify the converted values are within their allowable ranges
      1. Seed [0 – 10000]
      2. Number of random numbers [100 – 1000]

Note: If an error occurs while converting the string values or testing if the values are within range, display the appropriate error (refer to the sample output for examples).

   D. Generate the required number of random numbers
      Note: Each random number should be less than 100 (0 to 99)
      1. Count the occurrences of the generated numbers
         Note: Use an array to store the counts (don't store the random numbers, just the counts).
   E. Calculate statistics using the data stored in the array
      1. Min – Which number was generated the fewest times?
      2. Max – Which number was generated the most times?
      3. Average – On average, how many times was a number generated?
      4. Median – What was the median times a number was generated?
         Note: To correctly calculate the median value, first make a copy of the array, then sort the copied array, and finally, calculate the median value.
   F. Print a "Statistics" report
      1. Title the report "Random Number Statistics Report"
      2. List the calculated statistics (min, max, average, median)
      3. List the contents of the array
         a. Use the format POSITION : COUNT [PERCENTAGE%]
            Note: Show 1 decimal place for the PERCENTAGE value.
         b. The report should contain exactly 8 columns
            Note: The final row may not be a full row

Additional Notes

- Use one variable for each value (3 input/4 calculated/1 array = 8 variables)
- Create a constant for the size of the array (i.e., #define ARRAY_SIZE 100)
- Use "flag" variables as necessary
- Ensure your source code conforms to the programming standards for the class

Example Output

```
Input.txt file contents "10 1000" // Valid data
C:\Project4 Input.txt

Ima C Student
Project 4 - Random Number Statistics

Minimum Count: 0
Maximum Count: 19
Median Count: 10.00
Average Count: 10.00

Array analysis (Position: Count [Percentage])

  0: 9 [0.9%]   1:11 [1.1%]   2: 9 [0.9%]   3:10 [1.0%]   4: 7 [0.7%]   5: 8 [0.8%]   6:13 [1.3%]   7: 8 [0.8%]
  8:12 [1.2%]   9: 9 [0.9%]  10: 6 [0.6%]  11:12 [1.2%]  12: 8 [0.8%]  13:15 [1.5%]  14:10 [1.0%]  15:10 [1.0%]
 16:12 [1.2%]  17: 9 [0.9%]  18: 6 [0.6%]  19:12 [1.2%]  20:16 [1.6%]  21: 8 [0.8%]  22:10 [1.0%]  23:18 [1.8%]
 24:11 [1.1%]  25: 9 [0.9%]  26:10 [1.0%]  27: 6 [0.6%]  28:10 [1.0%]  29: 9 [0.9%]  30: 8 [0.8%]  31:10 [1.0%]
 32:10 [1.0%]  33: 5 [0.5%]  34:10 [1.0%]  35: 7 [0.7%]  36: 7 [0.7%]  37:13 [1.3%]  38: 8 [0.8%]  39:12 [1.2%]
 40: 4 [0.4%]  41: 4 [0.4%]  42: 8 [0.8%]  43:12 [1.2%]  44:14 [1.4%]  45:12 [1.2%]  46: 8 [0.8%]  47:11 [1.1%]
 48:11 [1.1%]  49: 9 [0.9%]  50:19 [1.9%]  51:10 [1.0%]  52: 9 [0.9%]  53:11 [1.1%]  54: 0 [0.0%]  55:16 [1.6%]
 56:10 [1.0%]  57: 9 [0.9%]  58: 8 [0.8%]  59:14 [1.4%]  60: 9 [0.9%]  61:10 [1.0%]  62:14 [1.4%]  63: 5 [0.5%]
 64:15 [1.5%]  65: 9 [0.9%]  66:10 [1.0%]  67: 8 [0.8%]  68: 7 [0.7%]  69: 8 [0.8%]  70:12 [1.2%]  71: 9 [0.9%]
 72:15 [1.5%]  73:11 [1.1%]  74: 7 [0.7%]  75: 6 [0.6%]  76: 7 [0.7%]  77:12 [1.2%]  78:15 [1.5%]  79:12 [1.2%]
 80: 8 [0.8%]  81: 5 [0.5%]  82: 6 [0.6%]  83:10 [1.0%]  84:11 [1.1%]  85: 4 [0.4%]  86:14 [1.4%]  87:11 [1.1%]
 88:15 [1.5%]  89: 8 [0.8%]  90:14 [1.4%]  91:13 [1.3%]  92:18 [1.8%]  93:11 [1.1%]  94:14 [1.4%]  95: 9 [0.9%]
 96: 7 [0.7%]  97: 6 [0.6%]  98: 8 [0.8%]  99:15 [1.5%]
```

```
Input.txt file contents "10 10" // The second data value is out of range
C:\Project4 Input.txt

Ima C Student
Project 4 - Random Number Statistics

The supplied value is out of range [100 - 1000]
```

```
Input.txt file contents "10 A" // The second data value is not numeric
C:\Project4 Input.txt

Ima C Student
Project 4 - Random Number Statistics

At least one of the supplied values is invalid.
```

```
C:\Project4 // No data file

Ima C Student
Project 4 - Random Number Statistics

The required filename parameter is missing.
```