```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#define PRINTF 0
//Thomas Gibbons

//prototypes for files.h
//{
int* readFile(int* sampleCount,int* sampleMax,char* filename);
double* offsetFile(int* sampleCount,int* sampleArray, double offset);
double* scaleFile(int* sampleCount,int* sampleArray, double scale);
void printArray(int Count,double* Array);
void outputFile(double count, double value, double* data, char* outputFile);
//}

//prototypes for calculations.h
//{
double mean(int* data,int count);
int maxValue(int* data, int count);
//}
int main()
{
    //get choice between 1 and 99 from user
    //{
        int inputFile=-1;
        while(inputFile<0 || inputFile>99)
        {
            printf("\nChoose data file 0-99: ");
            scanf("%d",&inputFile);
        }
    //}
    //creates string of filename user selects
    //{
        char* filename=malloc(15*sizeof(char));
        if(inputFile<10)
            sprintf(filename,"Raw_data_0%d.txt",inputFile);
        else
            sprintf(filename,"Raw_data_%d.txt",inputFile);
    //}
    //read data and store integers in array
    //{
        int Count, Max;
        int* Array=readFile(&Count,&Max,filename);
    //}
    //ends program if input file invalid
    //{
        if (Array==NULL)
        {
            printf("%s could not be accessed\n",filename);
            free(filename);
            return 1;
        }
    //}
```

```
        //get selection for offset or scale
        //{
            int choice=0;
            while(choice!=1 && choice !=2)
            {
                printf("\n1)Offset Data\n2)Scale Data\nChoose option: ");
                scanf("%d",&choice);
            }
        //}
        //same amount of space for both scaled and offset
            char* outFile=malloc(18*sizeof(char));

        if(choice==1)
        //operations for offsetting data
        {
            //receive offset value
                double value;
                printf("\nOffset value: ");
                scanf("%lf",&value);

            //creates string of output file name user selected
                double* offset=offsetFile(&Count,Array,value);
                if(inputFile<10)
                    sprintf(outFile,"Offset_data_0%d.txt",inputFile);
                else
                    sprintf(outFile,"Offset_data_%d.txt",inputFile);

            //print offsetted data to output file
                outputFile(Count,value,offset,outFile);

            //free memory allocated
                free(offset);
        }
        else
        //operations for scaling data
        {
            //receive scaling value
                double value;
                printf("\nScale value: ");
                scanf("%lf",&value);

            //creates string of output file name user selected
                double* scale=scaleFile(&Count,Array,value);
                if(inputFile<10)
                    sprintf(outFile,"Scaled_data_0%d.txt",inputFile);
                else
                    sprintf(outFile,"Scaled_data_%d.txt",inputFile);

            //print scaled data to output file
                outputFile(Count,value,scale,outFile);
            //free memory allocated
                free(scale);
        }
```

```
    //stats data output
    //{
        int maxData=maxValue(Array,Count);
        double ave=mean(Array,Count);
        char* statFile=malloc(22*sizeof(char));
        if(inputFile<10)
            sprintf(statFile,"Statistics_data_0%d.txt",inputFile);
        else
            sprintf(statFile,"Statistics_data_%d.txt",inputFile);

        outputFile(ave,maxData,NULL,statFile);
    //}
    //centered data output
    //{
        double* centered=offsetFile(&Count,Array,ave*-1);
        char* centeredFile=malloc(20*sizeof(char));
        if(inputFile<10)
            sprintf(centeredFile,"Centered_data_0%d.txt",inputFile);
        else
            sprintf(centeredFile,"Centered_data_%d.txt",inputFile);

        outputFile(Count,ave*-1,centered,centeredFile);
    //}
    //normalized data output
    //{
        double* normalized=scaleFile(&Count,Array,1.0/Max);
        char* normalizedFile=malloc(22*sizeof(char));
        if(inputFile<10)
            sprintf(normalizedFile,"Normalized_data_0%d.txt",inputFile);
        else
            sprintf(normalizedFile,"Normalized_data_%d.txt",inputFile);

        outputFile(Count,1.0/Max,normalized,normalizedFile);
    //}
    //free allocated memory
    //{
        free(centered);
        free(centeredFile);
        free(normalized);
        free(normalizedFile);
        free(statFile);
        free(outFile);
        free(filename);
        free(Array);
    //}
    //end succesfully
        printf("\n");
        return 0;
}
//functions for files.c
//{
int* readFile(int* sampleCount,int* sampleMax,char* filename)
```

```
/*  input:  address to store count
            address to store max value of data
            name of data file
    output: address of array of integer data*/
{
    FILE *fp;
    fp=fopen(filename,"r");

    if(fp==NULL)
        return NULL;

    fscanf(fp,"%d %d",sampleCount,sampleMax);
    int count=*(sampleCount);

    int* sampleArray;
    sampleArray=malloc(sizeof(int)*count);
    int x=0;

    while(count>0)
    {
        fscanf(fp,"%d", sampleArray+x);
        x++;
        count--;
    }

    fclose(fp);
    return sampleArray;
}

double* offsetFile(int* sampleCount,int* sampleArray, double offset)
/*  input:  address of count
            address of array of integer data
            value of offset
    output: address of array of double off-setted data*/
{
    double* offsetArray=malloc(*(sampleCount)*sizeof(double));
    int x=0;
    int count=*(sampleCount);

    while (count>0)
    {
        *(offsetArray+x)=*(sampleArray+x)+offset;
        x++;
        count--;
    }
    return offsetArray;
}

double* scaleFile(int* sampleCount,int* sampleArray, double scale)
/*  input:  address of count
            address of array of integer data
            value of scale
    output: address of array of double scaled data*/
```

```
{
    double* scaleArray=malloc(*(sampleCount)*sizeof(double));
    int x=0;
    int count=*(sampleCount);

    while (count>0)
    {
        *(scaleArray+x)=*(sampleArray+x)*scale;
        x++;
        count--;
    }
    return scaleArray;
}

void printArray(int Count,double* Array)
/*  input:  value of count
            address of array of double data
    output: displays double data*/
{
    int x=0;
    while(Count>0)
    {
        printf("%.4f ", (float)*(Array+x));
        x++;
        Count--;
    }
}

void outputFile(double count, double value, double* data, char* outputFile)
/*  input:  amount of data
            value(offset or scale) to be put in file
            double array to be printed
            name of file to output to*/
{
    FILE *write;

    write=fopen(outputFile,"w");

    fprintf(write,"%lf %lf\n",count, value);

    int x=0;
    if(data!=NULL)
        while(count>0)
        {
            fprintf(write,"%.4f\n",*(data+x));
            x++;
            count--;
        }

    fclose(write);

    printf("\n%s is loaded",outputFile);
}
```

```
//}

//functions for calculations.c
//{
double mean(int* data, int count)
/*  input:  integer array
            number of integers in array
    output: average of integers*/
{
    int total=0;
    int tempCount=count;

    while(tempCount>0)
    {
        total+=*(data+count-tempCount);
        tempCount--;
    }

    return (double) total/count;
}

int maxValue(int* data,int count)
/*  input:  integer array
            number of integers in array
    output: maximum value in array*/
{
    int tempCount=count;
    int maxValue=INT_MIN;
    while(tempCount>0)
    {
        maxValue=(maxValue>*(data+count-tempCount))? maxValue:*(data+count-tempCount);
        tempCount--;
    }

    return maxValue;
}


//}
```