Create a directory called **final**. In this directory, complete the program named **one.c** that reads integers from the file specified on the command line and inserts these values into a linked list (inserting at the end of the list). After building the list, the program prompts the user for a number **N**. The function **listPrint** then prints out every **Nth** number in the list.

You must write **listPrint**. The rest of the program, shown at the right, has been written for you and can be downloaded from Blackboard.

As an example, suppose the data file **data1** had the following values in it

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 30 40 50 90 99
```

A few sample executions of the program with that file are shown below:

```
$ ./a.out data1
Printing every Nth element
        Enter the print increment : 2
1 3 5 7 9 11 13 15 17 19 30 50 99

$ ./a.out data1
Printing every Nth element
        Enter the print increment : 5
1 6 11 16 30

$ ./a.out data1
Printing every Nth element
        Enter the print increment : 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 30 40 50 90 99

$ ./a.out data1
Printing every Nth element
        Enter the print increment : 11
1 12 50
```

```
// one.c can be downloaded from Blackboard

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

typedef struct node {
      int data;
      struct node *next;
} Node;

Node *add(Node *head, int e) {
   Node *newOne = (Node *) malloc (sizeof(Node));
   newOne->data = e;
   newOne->next = NULL;
   if (head == NULL) return newOne;
   Node *temp = head;
   while (temp->next != NULL) temp = temp->next;
   temp->next = newOne;
   return head;
}

void listPrint(Node *list, int inc) { return; }

int main(int argc, char *argv[]) {
   FILE *fp = fopen(argv[1], "r");;
   if (!fp) { printf("No such file\n"); exit(1); }
   int num;
   Node *theList = NULL;
   fscanf(fp, "%d", &num);
   while ( !feof(fp) ) {
      theList = add(theList, num);
      fscanf(fp, "%d", &num);
   }
   printf("Printing every Nth element\n");
   printf("\tEnter the print increment : ");
   scanf("%d", &num);
   listPrint(theList, num);
   fclose(fp);
   return 0;
}
```

Next, complete the program named **two.c** that uses two-dimensional arrays. The program keeps track of elevation data. Reading from the file specified on the command line, it first calls **readData** to read in a grid of elevation information. The then calls **printStats** and generates some basic statistics regarding this elevation data. The input file format is:

```
R (number-of-rows)
C (number-of-columns)
row0_column0_elevation … row0_column(C-1)_elevation
row1_column0_elevation … row1_column(C-1)_elevation
row(R-1)_column0_elevation … row(R-1)_column(C-1)_elevation
```

You must write **readData** and **printStats**. The rest of the program, shown at the right, can be downloaded from Blackboard.

- The function **readData** declares a Grid, reads in the number of rows and columns, allocates space for the data array, and then reads the data array.
- The function **printStats** generates the statistics shown below (average height, number of elevation points below this average, the number of elevation points equal to this average and the number of elevation points above this average.

As an example, executing this program with **data2a** and **data2b** yields:

data2a 3 4 10 20 30 40 20 30 40 50 90 90 90 90	**./a.out data2a** The average elevation is 50.000000 Points below this elevation = 7 Points equal to this elevation = 1 Points above this elevation = 4
data2b 2 3 1 2 3 4 5 60	**./a.out data2b** The average elevation is 12.500000 Points below this elevation = 5 Points equal to this elevation = 0 Points above this elevation = 1

```
// two.c can be downloaded from Blackboard

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

typedef struct grid {
     int rows;
     int columns;
     int **data;
} Grid;

Grid *readData(FILE *fp) {
   Grid *theGrid;
   return theGrid;
}

void printStats(Grid *myGrid) {
   return;
}

int main(int argc, char *argv[]) {
   Grid *myGrid;
   FILE *fp = fopen(argv[1], "r");
   if (!fp)
       { printf("No such file\n"); exit(1); }
   myGrid = readData(fp);
   printStats(myGrid);
   fclose(fp);
   return 0;
}
```

After completing both one.c and two.c, bundle your directory final into a single (compressed) zip file and submit it via Blackboard.