Lab 8: Pointers and dynamic memory allocation

Goals

Demonstrate proficiency in: (1) using the *malloc()* function to dynamically create an array; (2) using pointer notation to read values into the array; and (3) implementing a function that processes the array.

Background

Dynamic allocation of arrays

- The *void *malloc(int size)* (memory allocate) function is used to dynamically allocate an area of memory to be used as an array. The proper prototype for the *malloc()* function can be obtained by `#include <stdlib.h>`

- The value returned by *malloc()* is a (*void **) pointer to the block of memory that was allocated.

- The parameter passed to *malloc()* is the size of the area to be allocated *in bytes.* Therefore, if we want to allocate space for 100 *ints*, we must pass a value 400 to *malloc()* because each *int* occupies 4 bytes. However, the *sizeof()* facility should *always* be used instead of hard coding the constant 4.

- To dynmically create an array of 100 integers the statement:
  ```
  int *my_array = (int *) malloc(100 * sizeof(int));
  ```
  may be used. In the above statement the *cast* (int *) is used to keep *gcc* from issuing a *warning message* that the (*void **) pointer being returned by *malloc()* is being implicitly coverted to an (*int **) pointer.

- *A subscript is represented in pointer notation in the following way. To set element 13 of my_array to 99 the statement:*
  ```
  *(my_array + 13) = 99;
  ```
  may be used.

- *To read a value into element 13 use:*
  ```
  fscanf(stdin, "%d", my_array + 13);
  ```

Assignment:

Program p24.c on page 145 of the notes should be used as a starting point for this assignment. In a module called lab8.c write a *main()* function that *dynamically allocates* array of one hundred integers and then reads *an unknown number* of integer values into the array. (The number of values is guaranteed to be less than or equal to 100.) Use pointer NOT array notation in this function.

Now write a function:

*void istats(int *array, int count, int *min, int *max, int *avg);*

that is passed the array and the number of values read as input, and stores the minimum, maximum, and average values in the locations provided via the three pointers.

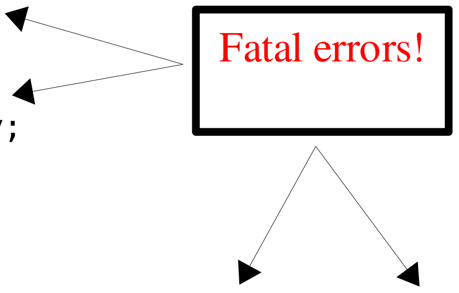
The main function should print (on 4 separate lines)

- the number of values read;
- the minimum value
- the maximum value
- the average of the values;

Strongly resist the urge to make the following fatal errors in the main() function:

```
int main()
{
    int *min;
    int *max;
    int *avg;
    int *array;
    int count;
     :
     :

    istats(array, count, min, max, avg);
```



Step 1:

Begin by writing a program that reads an unknown number of values into the dynamically allocated array and prints the number of values that were read.

When and only when you and your TA are convinced that this much is correct...

Step 2:

Add the *istats()* function and the additional output.

Any use of subscript instead of pointer notation will lead to a deduction.

Turn In Work

Show your TA that you completed the assignment. Then turn in your lab8.c program using the command:

sendlab.101.*section_number* 8 lab8.c