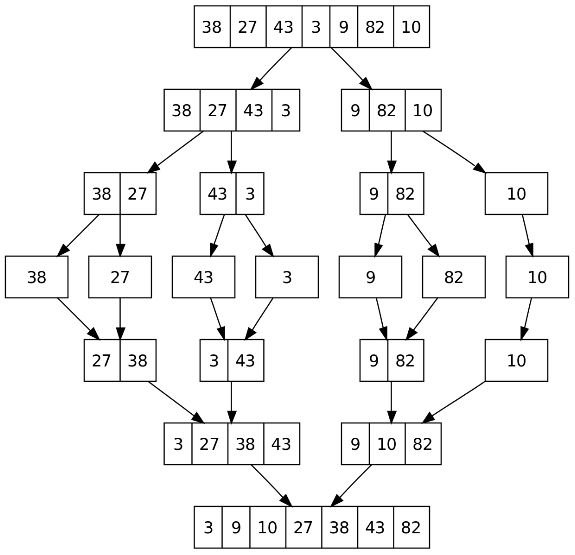
Timed Lab 4: Dynamic Memory

OBJECTIVE

In today's timed lab, you will be completing a program that takes in several positive integers from a user, sorts them, and then outputs the result. More specifically, the program will prompt the user for the number of elements, allocate the initially appropriate amount of memory, take the user's input, dynamically resize the array if the user gives it more input than expected, sort the array using merge sort, allocate memory for the sublists generated by merge sort, and finally return the sorted output. The input numbers are separated by spaces and the program stops taking input when the user enters a '0'.

You are **not** implementing the logic of merge sort and need no prior knowledge of merge sort to complete this lab. We have provided the bulk of merge sort's implementation. You only need to complete the code, in the flagged TODO locations, that involve memory allocation.

As a quick summary, the merge sort algorithm takes an unordered list, splits the list into two sublists, and then recursively splits those sublists until each contains one element. The one element sublists are then "merged," producing sorted sublists until the final sorted sublist remains. For your reference, this is a visual of how the algorithm works:



Your job is to write the code that:

1. Handles memory allocation for the primary array that holds the user input
2. Handles memory allocation for the sublist array(s) involved with merge sort

You will only need to write code in the areas marked with TODO. See the sample output below for the expected behavior of the program.

SAMPLE OUTPUT

Example #1:

```
> make merge_sort
> How many positive integers does your array contain? 5
> Enter an array of 5 positive integers (‘0’ to exit): 95 90 85 80 75 0
> Sorted: 75 80 85 90 95
```

Example #2:

```
> make merge_sort
> How many positive integers does your array contain? 5
> Enter an array of 5 positive integers (‘0’ to exit): 35 30 25 20 15 10 5 0
> You gave us an incorrect array size!
> Your array actually contained 7 items.
> Sorted: 5 10 15 20 25 30 35
```

RESTRICTIONS

- As always, your code must not crash, run infinitely, or produce **any memory leaks**
- Do NOT use realloc() in your code
- Your code must compile with the Makefile that we have provided! If it does not compile with our Makefile you will receive a 0

RUNNING YOUR CODE

We have provided a Makefile for this assignment that will build your project. Your code must compile with our Makefile. Here are the commands you should be using with this Makefile:

- To run the code: *make merge_sort*
- To run the code with valgrind: *make run-valgrind*

After you run valgrind, it should say "All heap blocks were freed -- no leaks are possible", otherwise you have a memory leak. valgrind should produce no other errors.

DELIVERABLES

Submit only your **merge_sort.c** to T-Square