Homework 4

Mahdiyeh Mirsharifi, Amin Parchami, Masih Majidi

Ammy (Emma’s other best friend) is implementing a merge sort algorithm for her midterm project. Emma suggests her to use an efficient assembly program for the **final merging step**. As always, she wants you to help her with the coding.

The program consists of four source files: “main.asm”, “merge.asm”, “io.asm”.

In “main.asm” you will invoke three functions readArray, merge, and printArray in order.

readArray: Takes a single input parameter which is the address of an array of double words. It keeps reading positive integers using **read_int** and storing in consecutive elements of the array until a nonpositive (<= 0) input is entered. The function must return the number of the array elements received from the input as its return value (the final nonnegative element is not stored in the array).

merge: Takes 6 input parameters. The first 3 are (the addresses of) 3 arrays (call them L1, L2 and L3). The second 3 are integers storing certain indices of L1, L2 and L3 respectively. The function compares **`L1[i]`** and **`L2[j]`**. If **`L1[i] <= L2[j]`** you must set **`L3[k] <- L1[i]`** and return zero as the return value. Otherwise set **`L3[k] <- L2[j]`** and return 1 as the return value.

printArray: Takes (the address of) an array and the array length as input parameters and prints the array in a space-separated format.

Assume that the input array elements are entered ascendingly. The functions readArray and printArray must be written in the file io.asm and the function merge must be written in the file merge.asm.

In “main.asm”, allocate space for the three arrays in the .data or .bss sections. Assume that the inputs are at most 1000 elements large. Use “readArray” twice to get the input sequences. Then you will use “merge” **as many times as needed** to merge the input arrays and create the resulting sorted array. Finally, print the result using “printArray”.

You can find a script named **test.sh** attached to the homework. It checks whether your functions (in merge.asm and io.asm) comply with the above requirements. Just call the script by running "./test.sh" or "bash test.sh" and check the output.

You also need to write a **Makefile** which assembles and links your three **.asm** files.

Your code **must** comply with the following rules:

- You must use the memory/data segment (or BSS segment).
- You must use the read_int, print_int (from the textbook) for I/O.
- You can only use the commands you have learned so far in the class.

Please notice that your code will be checked for similarity. In the case of cheating the student will receive a negative point. It is your responsibility to protect your own code.

Please upload only the four “**.asm**” files plus the **Makefile** on courses.kntu.ac.ir.

Example:

Input:

```
1 4 6 7 9 0
2 3 8 12 23 0
```

Output:

```
1 2 3 4 6 7 8 9 12 23
```