CS311 COMPUTER ORGANIZATION
HOMEWORK 2

NGUYEN BA VINH QUANG – 20190710

A. STACK ALLOCATION LAYOUT FOR EACH PROCEDURE

1. **"quicksort" procedure**

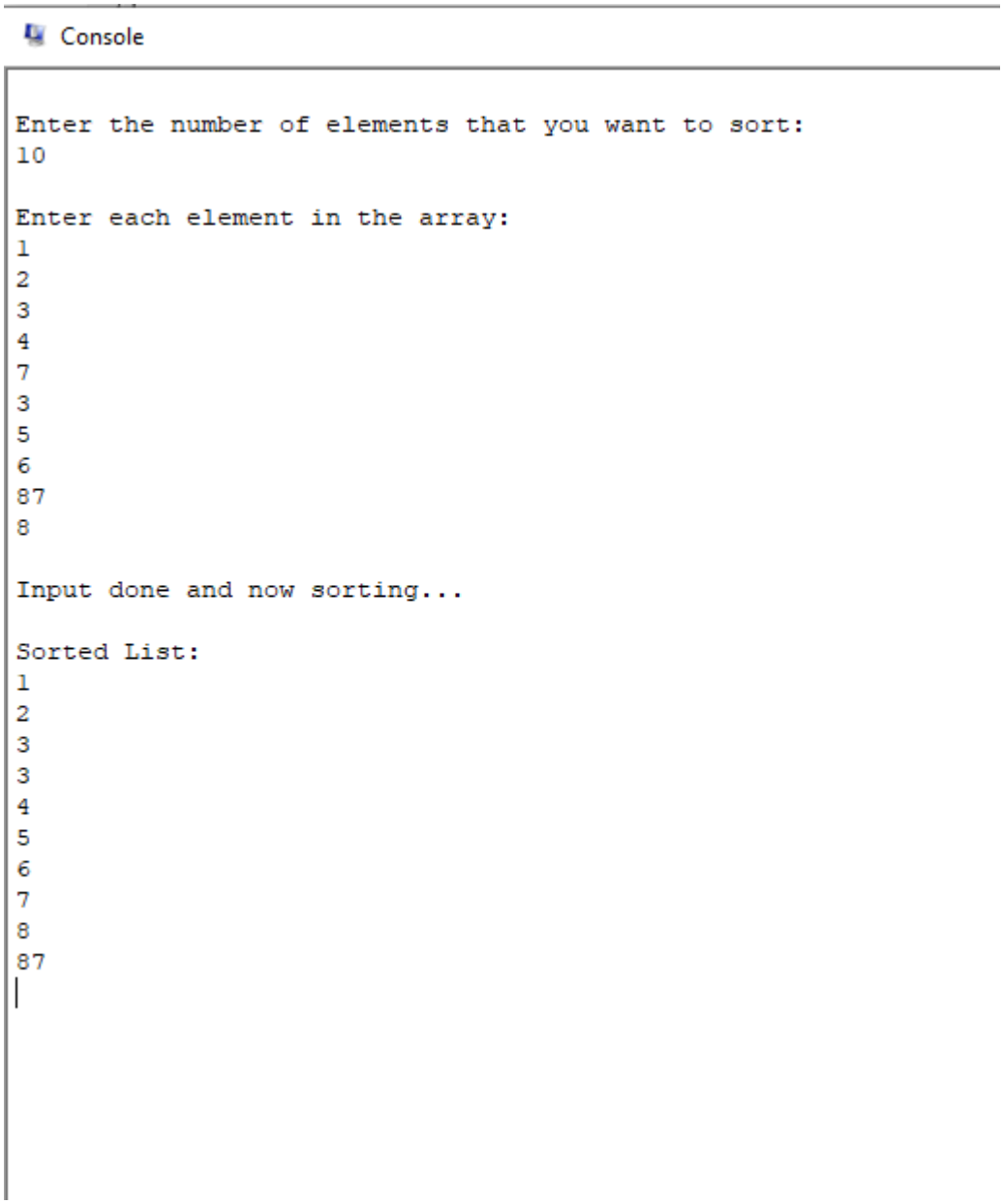
   In this procedure, I used stack space to store 3 values at each time the recursive was called: low in 0($sp), high in in 4($sp) and $ra in 8($sp).

2. **"partition" procedure**

   In this procedure, I used stack space to store only the $ra in 0($sp).

B. BRIEF EXPLANATION ON IMPLEMENTATION

1. **The consideration before implementation**
   - In MIPS, I can not call the recursion as in high level programming languages. It means the stack is not autonomous.
   - So, I need to handle the stack manually. Besides that, I need to consider which values should be stored in the stack.
   - I also have to find a way to read and store an array with the size upto 10^5.
2. **Implementation issues and dealing with them**
   - The first issue is to find a way to read and store the array. I used a ".space 400004" to take 400004 consecutive bytes in the memory. Why is it 400004? Because each word contains 4 bytes and we need 100000 at most, and for convenience purposes, I will not use the 4 first bytes of the space.
   - The second issue that I had to deal with is the $ra register, it took me a large amount of time to understanding deeply what is going on with the "jr $ra".
   - The third issue is the equation "i = low + (1664525*(unsigned)high + 22695477*(unsigned)low)%(high-low+1);". The numbers here can exceed the limit of a 4 bytes integer, so I broke this equation into some parts using the properties of modulo, such as (1664525%(high-low+1))*(high%(high-low+1)), etc…
   - Finally, sometimes, I forgot to delete the values from the stack after the recursion was call.



This is an example of Simulator Console when I run my program.