Assignment 3

Q1. Write a function called “merge”, which takes three integer arrays as arguments; there will actually be five arguments (each 32-bit): array pointer 1, array length 1, array pointer 2, array length 2, and result array pointer. The routine assumes that the input arrays are sorted, and merges the input arrays into the result array. The function can assume that the result array has enough space.

Your main function should input an array of 8 integers onto an array (you can copy code from the previous exercise). Call merge on the two halves of the 8-element array. Don't forget to declare another 8-element array in your data segment, for the output. The main routine should also print the output array.

Q2. Write a code for the assembly version of the recursive factorial computation. Your main program should call the factorial routine on an integer.

Q3. Use the merge procedure from Lab 2, and write a recursive merge-sort procedure. The recursive procedure should take two arguments: the starting address of the sub-array to be sorted, and the length of this sub-array. The procedure, before it returns, should guarantee that the (sub-)array given as argument is sorted.
Hint-1: You can use a global temporary array for the merging, after the recursive calls, and copy over from the temporary array to the relevant portion of the main array.
Hint-2: Your main routine should input an 8-element integer array from the user, and call the merge-sort procedure to sort it.

Q4. At what point in your tree of call activations will the stack depth be the maximum (just with respect to the recursive calls)? Insert a breakpoint at a place in the code where the stack is at its maximum size. Examine the stack and explain its contents with reference to your code. Point out the various call activation records.
Hint: you have to insert a breakpoint at an appropriate place in the mergesort procedure, not the merge procedure.