1. (1) **Write a program** to copy a string of characters from STRING to memory space starting from address 0x40000050, **reversing the string** in the process. Assume the program has the following declaration initially.

   STRING DCB "THE SOURCE STRING", 0

   (2) **Write a program** to copy the integers from TABLE to memory space starting from address 0x4000000, **reversing the integer order** in the process. Assume the program has the following declaration initially.

   TABLE DCD 0x00000013, 0x80808080, 0xFFFF0000, 0x1111FFFF
   DCD 0xFEBBAAAA, 0x12340000, 0x88881111, 0x22227777

   (3) **Write a program** to **reverse** the bits in R0 so that the register R0 containing $b_{31}b_{30}b_{29}\ldots b_1b_0$ now contains $b_0b_1\ldots b_{29}b_{30}b_{31}$, assuming R0 contains the value 0xDEADFACE initially.

2. Assume the word at memory address 0x40000000 = 0x6543CDEF, the word at memory address 0x40000004 = 0xABCD8765, the word at memory address 0x40000008 = 0x12345678, the word at memory address 0x4000000C = 0xBEEFFACE. **Write a program** that includes **6 subroutines func1**, **func2a, fun2b**, **func3**, **func4**, **func5** and calls to the 6 subroutines one by one.

   (1) **func1** uses EOR to put even parity for bits 3, 4, 5, 9 and 11 of the word at memory address 0x40000000 into bit 8 of the word at memory address 0x40000004.

   (2) **func2a and func2b** put odd parity for bits 2, 4, 6 and 8 of the word at memory address 0x40000000 into bit 9 of the word at memory address 0x40000004.
   (a) use EOR.
   (b) use TST and do not use EOR.

   (3) **func3** puts bits 30, 29, 28, 27, 15, 14, 13, 10 of the word at memory address 0x40000000 into bits 23~16 of the word at memory address 0x40000004 respectively,

   (4) **func4** puts the number of different bits between the word at memory address 0x40000008 and the word at memory address 0x4000000C into the word at memory address 0x40000010.

   (5) **func5** puts the different bit numbers between the word at memory address 0x40000008 and the word at memory address 0x4000000C one by one in the words started from memory address 0x40000070.

   (Do not forget to use **STMDB** with **SP!** and the corresponding **LDM** in each subroutine to avoid the side effect, if any, and **highlight the memory locations with the related registers stored using STM and the registers restored using LDM**. **Assume SP = 0x40000020 initially.**)

3. Assume the values in memory addresses 0x40000040, 0x40000044, 0x40000048, 0x4000004C are respectively 0x87654321, 0x12345678,, 0xFACEBEEF, 0xBEEFFACE.
   Write a program in the following 8 steps
   (1) Use LDMIA (2) Use LDMIB (3) Use LDMDA (4) Use LDMDB
   to load the values in addresses 0x40000040, 0x40000044, 0x40000048, 0x4000004C into
   (1) r0, r1, r2, r3 respectively.
   (2) r1, r2, r3, r4 respectively.
   (3) r2, r3, r4, r5 respectively.
   (4) r3, r4, r5, r6 respectively.
   (5) Use STMIA (6) Use STMIB (7) Use STMDA (8) Use STMDB
   to store r0, r1, r2, r3 into memory with
   (5) addresses 0x40000058, 0x4000005C, 0x40000060, 0x40000064 respectively.
   (6) addresses 0x40000068, 0x4000006C, 0x40000070, 0x40000074 respectively.
   (7) addresses 0x40000078, 0x4000007C, 0x40000080, 0x40000084 respectively.
   (8) addresses 0x40000088, 0x4000008C, 0x40000090, 0x40000094 respectively

Note: Please
   (1) put necessary **Keil Tool DEBUG window screenshots** to show your **program** and **execution results** including **highlighted necessary initial assumptions and subsequent memory and register changes**,
   (2) **comment student ID+your English name in every screenshots**, and
   (3) put reports into one word file named by student_ID+your_name.