```
                AREA INSERT_SORT_PGM, CODE, READONLY
                ENTRY
MAIN            ADR sp, STACK_START




STOP            B STOP
END_MAIN

INSERTION_SORT









END_INSERTION_SORT

INSERT









END_INSERT

                AREA INSERT_SORT_PGM, DATA, READWRITE

SCORES          DCD 20, 30, 10, 40, 50, 60, 30, 25, 10, 5
N               DCD 10
STACK_END       SPACE 0x00000FF
                ALIGN
STACK_START     DCD 0
DUMMY           DCD    0x12345678
                END
```

```
main:
integer scores [100];
integer n;  // # of elements

InsertionSort(scores, n)
(*)

end main




InsertionSort(numbers - addr. to array,
              length - integer)

 integer firstUnsortedIndex
 for firstUnsortedIndex=1 to (length-1) do
     Insert(numbers,
            numbers[firstUnsortedIndex],
            firstUnsortedIndex-1);

 end for
end InsertionSort









Insert(numbers - address to integer array,
       elementToInsert - integer,
       lastSortedIndex - integer) {
   integer testIndex;
testIndex = lastSortedIndex;
while (testIndex >=0) AND
      (numbers[testIndex] > elementToInsert ) do
   numbers[ testIndex+1 ] = numbers[ testIndex ];
   testIndex =  testIndex - 1;
end while
numbers[ testIndex + 1 ] = elementToInsert;

end Insert
```

2. If r5 contains 0xAF000A5D and r6 contained 0x6, what hexadecimal value would be in r4 after each of the following?

a) MOV r4, r5, LSL #3

b) MOV r4, r5, LSR #6

c) MOV r4, r5, ASR #3

d) MOV r4, r5, ROR #3

3. If r5 contains 0xA5D and r6 contained 0x63C, what hexadecimal value would be in r4 after each of the following?

a) AND r4, r5, r6

b) ORR r4, r5, #0xBF

c) EOR r4, r5, r6

d) BIC r4, r5, r6

e) MOVN r4, r5

4. Sometimes you want to manipulate individual bits in a "*string of bits*". For example, you can represent a set of letters using a bit-string. Each bit in the bit-string is associated with a letter: bit position 0 with 'A', bit position 1 with 'B', ..., bit position 25 with 'Z'. Bit-string bits are set to '1' to indicate that their corresponding letters are in the set, and '0' if not in the set. For example, the set { 'A', 'B', 'D', 'Y' } would be represented as:

	unused	'Z'	'Y'	'X'	. . .	'E'	'D'	'C'	'B'	'A'
{ 'A', 'B', 'D', 'Y' } is	0 0 0 0 0 0	0	1	0		0	1	0	1	1
bit position:		25	24	23		4	3	2	1	0

To determine if a specific ASCII character, say 'C' (67_{10}) is in the set, you would need to build a "mask" containing a single "1" in bit position 2.

a) What instruction(s) could we use to build the mask needed for 'C' in r3?

b) If a bit-string set of letters is in register r5, then what instruction(s) can be used to check if the character 'C' (using the mask in r3) is in the set contained in r5?

c) If a bit-string set of letters is in register r5 and another in r6, then what instruction(s) can be used to calculate in r7 the union of sets in r5 and r6? (i.e., all elements in either set)

d) If a bit-string set of letters is in register r5 and another in r6, then what instruction(s) can be used to calculate in r7 the intersection of sets in r5 and r6? (i.e., all elements in both sets)

e) If a bit-string set of letters is in register r5 and another in r6, then what instruction(s) can be used to calculate in r7 the set difference: set in r5 - set in r6? (i.e., all elements in the left-hand set that are not in the right-hand set)