You should attempt as many questions as you can before the examples class.
You can discuss the questions with other students, but remember that the exam may contain similar
questions, so make sure you can answer them by yourself.

Make sure that you can answer all the exercises given in the lectures
If there is anything that you don’t completely understand, try to sort it out now.

1.

```
ADRL R1, x
LDR  R2, [R1]
LDR  R3, [R1, #4]
ADD  R2, R3, R2
LDR  R3, [R1, #8]
ADD  R2, R3, R2
LDR  R3, [R1, #12]
ADD  R2, R3, R2
SWI  2
x   DEFW 4
   DEFW 3
   DEFW 1
   DEFW 5
```

a) What is the value of R2 after the above program has executed?
b) Rewrite the program using post-indexed addressing
c) Rewrite the program using post-indexed addressing within a loop. (Use the fact that you know
   how many numbers are in the list to control the loop.)

2. Write methods that:
   are given as parameters
   – the address of a list of numbers in R1
   – the number of numbers in R0
   and contain a similar loop to your answer to 1c above
   but return the answer in R0.

   Write the methods:
   a) Assuming the Link Register need not be stacked.
   b) Using the stack to remember the Link Register.

3. Rewrite the following Java expressions in ARM code, assuming w, x, y, z are all boolean variables
   and have been given memory locations using “DEFB” commands:

   a)  z= (x & y) | (!w);

   b)  z= (x &k y) || (!w);

   c)  z= x != y;

   Question 4 is on the next page
4. Encode the following Java switch statement in ARM assembler. You can assume that all variables are declared using “DEFW” or held in registers.

```java
int value;
int power;
// assume there is some code here that sets value and power
int valueToThePower1 = value;
int valueToThePower2 = valueToThePower1 * valueToThePower1;
int valueToThePower4 = valueToThePower2 * valueToThePower2;
int valueToThePower8 = valueToThePower4 * valueToThePower4;
int result = 1;
switch (power)
{
    case 10: result *= valueToThePower1;
    case 9: result *= valueToThePower1;
    case 8: result *= valueToThePower8;
        break;
    case 7: result *= valueToThePower1;
    case 6: result *= valueToThePower1;
    case 5: result *= valueToThePower1;
    case 4: result *= valueToThePower4;
        break;
    case 3: result *= valueToThePower1;
    case 2: result *= valueToThePower2;
        break;
    case 1: result *= valueToThePower1;
        break;
}
```