PRIFYSGOL ABERYSTWYTH UNIVERSITY

SEMESTER TWO 2025–2026

COMPUTER SCIENCE

CS10720 : PROBLEMS AND SOLUTIONS

Time allowed: **2 HOURS**

Answer all **THREE** questions

All questions carry equal marks.

YOU MAY BRING THE FOLLOWING INTO THE EXAMINATION:
UP TO 3 SHEETS OF NOTES ON A4 PAPER (WHICH MAY BE DOUBLE-SIDED). THERE ARE NO ADDITIONAL RESTRICTIONS ON CONTENT OR FORMATTING OF THESE NOTES.
NO ELECTRONIC DEVICES.

CALCULATORS ARE NOT PERMITTED.

STUDENTS MAY SUBMIT ANSWERS TO THIS PAPER IN EITHER WELSH OR ENGLISH

Once seated, students may complete the front cover of the answer book(s) and the attendance slip ONLY.

DO NOT TURN THE EXAMINATION QUESTION PAPER OVER UNTIL INSTRUCTED TO DO SO.

1. **Numbers and Matrices**

a) **Converting integers from decimal to binary representation**

You are asked to convert the decimal number -14 into different binary representations, always using 5 bits. If a number cannot be represented using 5 bits write 'can't be represented' as answer. You are also asked to convert the bit string 10110 into decimal representation, for different binary representations.

To make it clear what part of your answer refers to what number and what representation, reproduce the following table in your answer book and fill in the empty cells. Fill the first empty column with the appropriate binary representations (using 5 bits). Fill the second empty column with the appropriate decimal values.

Representation	-14	10110
signed magnitude		
one's complement		
two's complement		
excess (bias $b = 15$)		
excess (bias $b = 12$)		

[11 marks]

b) **Comparing numbers in binary IEEE 754 representation**

Describe an algorithm that receives as input the three components of an IEEE number in binary representation for two numbers (for number a, $\texttt{sign}_a$, $\texttt{exponent}_a$, and $\texttt{significant}_a$, and for number b, $\texttt{sign}_b$, $\texttt{exponent}_b$, and $\texttt{significant}_b$; each in form of an array). The output is -1 if $a < b$, 0 if $a = b$, 1 if $a > b$, and 'error' if a and b cannot be compared.

Describe your algorithm for general representation length (where the length of exponent and significant is available as `exponent.length` and `significant.length`. You should describe your algorithm in Welsh or English. There is no need to use actual Java code.

Analyse the runtime of your algorithm in the best and worst case, using asymptotic notation. [11 marks]

c) **Matrix multiplication**

Compute the result of the following matrix multiplication. It is sufficient to show your workings, there is no need to compute the actual result. For example, if $19 \times 12 + 3 \times (-6)$ would be part of a solution you would get marks for writing this. Computing 210 as result would not get you any additional marks. [11 marks]

$$\begin{pmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{pmatrix} \cdot \begin{pmatrix} 1 & 3 \\ 5 & 7 \\ 9 & 11 \\ 13 & 15 \end{pmatrix}$$

2. Turing machines, Computational Problems, Computability, and Reductions

a) **Halting problem**

Someone implements a Turing machine simulator on an Arduino. They argue that the halting problem for this simulator is computable because the Arduino only has 2 KB memory that can be used for the Turing machine itself and the memory it can use.

A friend agrees that it can be decided if a Turing machine halts for a given input for this simulator but says the reasoning is flawed in the following way. The halting problem takes as input programs, not Turing machines. The friend argues that while the problem of deciding if a Turing machine on the Arduino halts is computable, this is not related to the limited memory but caused by only looking at Turing machines.

First describe the halting problem and its properties. Then discuss if the original statement is true and, finally, discuss if you agree with your friend. [11 marks]

b) **Special cases and reductions**

A friend claims that if you have two computational problems X and Y, and you know that X is a special case of Y, then you can always reduce X to Y, but it might be the case that you cannot reduce Y to X: sometimes you can and sometimes you can't.

Do you agree? Give reasons for your answer. [11 marks]

c) **Special cases, reductions, computability**

Consider three computational problems A, B and C. Let us assume that we know the following three things to be true. (1) A is a special case of B. (2) B can be reduced to C. (3) C is not computable.

For all four possible combinations regarding the two problems and computability ((1) both computable; (2) A computable, B not computable; (3) A not computable, B computable, (4) both not computable), decide if this is possible or not. Give reasons for your answers.

What changes if you learn that additionally C can be reduced to A? [11 marks]

3. Algorithms and Runtime Analysis

a) **Quick sort**

We want to apply partition (from quick sort) to the array below. Write down the contents of the array after `int pt = partition(keys, 0, 9, 4)` is executed. What will the value of `pt` be at the end?

keys:	8	7	9	6	5	2	1	4	0	3
	0	1	2	3	4	5	6	7	8	9

Below is the source code of `partition`, provided as a courtesy.

```
1  int partition(int []keys, int left, int right, int pivot) {
2    int swap;
3    while ( left < right ) {
4      while ( keys[left] < keys[pivot] )
5        left++;
6      while ( keys[right] > keys[pivot] )
7        right--;
8      swap = keys[left];
9      keys[left] = keys[right];
10     keys[right] = swap;
11     if ( left == pivot )
12       pivot=right;
13     else if ( right==pivot )
14       pivot=left;
15     if ( left<pivot )
16       left++;
17     if ( right>pivot )
18       right--;
19   }
20   return pivot;
21 }
```

[11 marks]

b) **Runtimes of algorithms**

Assume you have two algorithms A_1 and A_2, both solving the same problem. The first has runtime $T_1(n) = 2n^2 + n + 2$ for an input of size n, the other runtime $T_2(n) = 81n + 1002$. Use Theta notation to provide the asymptotic runtimes of A_1 and A_2. Which algorithm is asymptotically faster? Which one is faster for input size $n = 1$? For what input size n does the asymptotically faster algorithm become actually faster? [11 marks]

c) **Algorithm design**

Consider the following computational problem. Input is an integer array of length n where n is an odd integer. Output are $\lfloor\sqrt{n}\rfloor$ specific numbers from the array: among all numbers in the array that are greater than the median number the output consists of the $\lfloor\sqrt{n}\rfloor$ smallest ones.

For example, consider an array of size 9 that contains the numbers 14, 18, 12, 10, 4, 8, 16, 6, 2, the median is 10. The numbers 12, 14, 16, 18 are greater than the median. The $\lfloor\sqrt{9}\rfloor = 3$ smallest of those are 12, 14, 16. Therefore, 12, 14, 16 is a correct output (but 16, 12, 14 would be equally valid as output).

Describe an efficient algorithm that solves this problem. Your description should be precise but there is no need to provide actual code. Your algorithm is allowed to change the contents of the array. We are only interested in obtaining the correct output at the end.

You can use algorithms that we have discussed in this module as subroutines. If you do this there is no need to explain the algorithm that you use. It is sufficient to name such algorithms and explain how you use them as part of your algorithm.

What is the runtime of your algorithm? We use the length of the array n as measure for the input size. Use asymptotic notation for the runtime and explain why your algorithm has this runtime. [11 marks]

END OF PAPER