ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Spring Semester 2021 Prof. R. Wattenhofer

Computational Thinking

Spring Semester 2021

Monday, August 30, 2021, 9:00-11:00

Do not open before the official start of the exam!

The exam lasts for 120 minutes and comprises 120 points. There is one block of questions for each lecture chapter.

You can answer in German or English, or mix German and English.

Unless explicitly stated, you do **not** have to justify your answers. However, writing down your thoughts (math, text or annotated sketches) will help us to understand your approach. This will allow us to award points even if your solution is wrong. Please make sure your writing is readable.

Some questions ask you to fill in answers in a template. In case you made a mess, and want a fresh start, you find fill-in replacements at the end of the exam.

Please write your name and student number on every extra sheet. Please write your name and student number in the following fields on this cover sheet.

Family Name	**First Name**	**Student Number**

Task	**Achieved Points**	**Maximum Points**
1 - Algorithms		15
2 - Complexity		17
3 - Cryptography		15
4 - Data and Storage		18
5 - Machine Learning		18
6 - Neural Networks		19
7 - Computability		18
Total		**120**

1 Algorithms (15 points)

Choose One

a) `[2 points]` Which of the following statements about recursion is *wrong*?

- ○ Recursion trades off description for time.
- ○ Recursive algorithms use the stack data structure by default.
- ○ Two functions cannot call each other recursively.
- ○ Recursive algorithms can always be replaced with non-recursive algorithms.

Fibonacci Sequence

The Fibonacci sequence starts with 0 and 1, and every next number is generated by adding the previous 2 numbers. The sequence goes like this: $0, 1, 1, 2, 3, 5, 8, 13, \ldots$

A simple recursive algorithm to generate the n^{th} Fibonacci number is below:

```
# Assume input n to be a non-negative integer.
def fibo(n):
    if n <= 1: return n
    return fibo(n-1) + fibo(n-2)
```

a) `[5 points]` The above approach calls `fibo` multiple times for the same input and can be optimized with memoization. Fill in the areas indicated by the comments (below) to implement memoization.

```
# Assume input n to be a non-negative integer.
def fibo2(n, memo = None):
    if not memo: memo = {0: 1, 1: 1}
    ##### Fill below #####


    ######################
    answer = fibo2(n-1, memo) + fibo2(n-2, memo)
    ##### Fill below #####


    ######################
    return answer
```

b) `[8 points]` The recursive algorithm with memoization is still inefficient with respect to space complexity. Can we get constant space complexity? If so, design such an algorithm. If not, explain why.

Multiple Choice

a) "Two functions cannot call each other recursively." is false.

Fibonacci Sequence

a) Completed Memoized Solution:

```
# Assume input n to be a non-negative integer.
def fibo2(n, memo = None):
    if not memo: memo = {0: 1, 1: 1}
    if n in memo: return memo[n]
    answer = fibo2(n-1, memo) + fibo2(n-2, memo)
    memo[n] = answer
    return answer
```

b) Constant Memory Solution:

```
# Assume input n to be a non-negative integer.
def fibo3(n):
    if n < 2: return n
    t1 = 0
    t2 = 1
    for i in range(n-1):
        next = t1 + t2
        t1 = t2
        t2 = next
    return (next)
```

2 Complexity (17 points)

Choose One

a) `[4 points]` Among the four decision problems A_1, A_2, B_1, B_2, we know that A_1 is in P and B_1 is NP-hard. The existence of which of the following polynomial time reductions implies that A_2 is also in P and B_2 is also NP-hard?

- ○ A reduction from A_1 to A_2 and a reduction from B_1 to B_2.
- ○ A reduction from A_1 to A_2 and a reduction from B_2 to B_1.
- ○ A reduction from A_2 to A_1 and a reduction from B_1 to B_2.
- ○ A reduction from A_2 to A_1 and a reduction from B_2 to B_1.

b) `[4 points]` Consider the OnePositive-SAT problem. This is a SAT-variant, where in each clause, there is exactly one positive literal, and the rest are negative. Assuming that the input has already been read (so it takes no time), how much time does the best possible algorithm need to decide if such a formula is satisfiable?

- ○ Constant time.
- ○ Not constant, but polynomial time.
- ○ Not polynomial, but exponential time.
- ○ The problem is undecidable.

Bin Packing

Recall the FirstFit approximation algorithm for Bin Packing.

```
1 def FirstFit(items, B):
2     for each item in items:
3       place item in the first bin where it still fits
4       if item does not fit into any bin:
5         open a new bin, and insert item into the new bin
```

In the lecture, we proved that FirstFit gives a 2-approximation. However, it was not shown that the algorithm is no better than this.

a) [9 points] Give an example of a sequence of items for which FirstFit uses c-times more bins than the optimum for some $c > 1$. Simply state the example as well as the number of bins for the optimum and for FirstFit. (The larger the c, the more points you get.)

Solution

Multiple Choice

a) A reduction from A_2 to A_1 and a reduction from B_1 to B_2.

b) Constant time - you just need to output Yes, since such a formula is always satisfiable, for example, simply all variables being True works.

Bin Packing

Let the size of the bins be 1. Proving $c \geq 3/2$ already gives full points: For the sequence $1/3, 1/3, 2/3, 2/3$, FirstFit uses 3 bins, while OPT only needs 2.

[A better lower bound of $c \geq 5/3$ can be shown using the sequentce $6 \times (1/7 + \epsilon), 6 \times (1/2 + \epsilon), 6 \times (1/3 + \epsilon)$. FirstFit uses $1 + 3 + 6 = 10$ bins while $OPT = 6$.]

3 Cryptography (15 points)

Bob wants to send money to the bank account of Alice. To do this, first Alice sends the details of her bank account (m) to Bob by using the following protocol, where k_p^A, k_p^B are respectively Alice's and Bob's public keys and k_s^A, k_s^B are their secret keys.

Protocol 1

Alice		**Bob**
has (k_p^A, k_s^A), knows k_p^B		has (k_p^B, k_s^B), knows k_p^A
$c =$`encrypt(`m`,` k_p^B`)`	send over c $\longrightarrow$	$m =$`decrypt(c,` k_s^B`)`
		Transfer the money to the given bank account m

a) `[6 points]` Describe an attack how Eve (man in the middle) can steal Bob's money in the given Protocol 1.

b) `[6 points]` Modify the protocol in the following table to fix the problem.

Protocol 2

Alice	**Bob**
has (k_p^A, k_s^A), knows k_p^B	has (k_p^B, k_s^B), knows k_p^A

$\longrightarrow$

c) `[3 points]` Alice and Bob need to know the public keys of each other (k_p^B and k_p^A) before they start Protocol 1 and 2. Which of the following methods can be used for this purpose?

- ○ Alice and Bob can simply exchange their public keys over Internet in plaintext, since public keys are anyway public information.
- ○ Alice and Bob can register at a certificate authority to get a certificate that binds their names with the public keys and exchange the certificates afterwards.
- ○ Alice and Bob can use the Diffie-Hellman Key Exchange algorithm.
- ○ It is not possible to do this without meeting in person.

Solution

a) [6 points] Protocol 1 offers secrecy but no authentication. Eve (in the middle) can simply encrypt her bank account information $c' = \texttt{encrypt}(m', k_p^B)$ and replace c with c'. Bob then will send the money to Eve instead of Alice.

b) [6 points] We can use digital signatures to achieve authentication as shown in the following protocol.

Protocol 2

Alice		**Bob**
has (k_p^A, k_s^A), knows k_p^B		has (k_p^B, k_s^B), knows k_p^A
$c = \texttt{encrypt}(m, k_p^B)$		$m = \texttt{decrypt}(c, k_s^B)$
$s = \texttt{sign}(c, k_s^A)$	send over c, s $\longrightarrow$	$b = \texttt{verify}(s, c, k_p^A)$
		Transfer the money to the given bank account m, if verification succeeds.

c) [3 points] Alice and Bob can register at a certificate authority to get a certificate that binds their names with the public keys and exchange the certificates afterwards.

4 Data and Storage (18 points)

Choose One

a) [3 points] Which statement about JOIN variants is correct?

- ○ A query with RIGHT OUTER JOIN can return fewer rows than the same query with INNER JOIN.
- ○ Any query with INNER JOIN can be rewritten to an equivalent query with RIGHT OUTER JOIN and a WHERE condition.
- ○ FULL OUTER JOIN and CROSS JOIN give the same result.
- ○ CROSS JOIN can be used to emulate all other JOIN variants.

Online Store

We consider the SQL database of an online shop. The database contains information about customers and products in these tables (table keys are underlined):

customers(customer_id, firstname, lastname, address)
orders(order_id, customer_id, order_status)
order_items(order_id, product_id, quantity)
products(product_id, product_name, price)

b) [3 points] Explain the purpose of the table `order_items`. Why does it exist in addition to the `orders` table?

c) [4 points] Explain what the following SQL query returns:

```
SELECT p.*
FROM products AS p
INNER JOIN order_items AS oi ON p.product_id = oi.product_id
GROUP BY p.product_id
ORDER BY SUM(oi.quantity) DESC
LIMIT 1;
```

d) `[8 points]` Write an SQL query that for each customer outputs the number of different products he has bought.

Solution

Multiple Choice

a) An INNER JOIN can be replaced with a RIGHT OUTER JOIN with WHERE (column from left table) IS NOT NULL. (A RIGHT OUTER JOIN contains all rows from an INNER JOIN and additionally those where the left table has no corresponding entries. CROSS JOIN forms the Cartesian product, rows with NULL entries as in OUTER JOINs do not arise)

Online Store

a) This table allows each order to have multiple products and vice-versa.

b) This query finds the most purchased product.

c) The following query finds the number of different products each customer has bought:

```
SELECT c.*, COUNT(DISTINCT oi.product_id) AS product_count
FROM customers AS c
LEFT OUTER JOIN orders AS o ON c.customer_id = o.customer_id
LEFT OUTER JOIN order_items AS oi ON o.order_id = oi.order_id
GROUP BY c.customer_id;
```

5 Machine Learning (18 points)

Choose One

a) `[4 points]` Look at the data you collected in Figure 1. You decide to fit a polynomial of degree 3. You repeat this many times (collect data, fit data, repeat). What do you expect from your model?

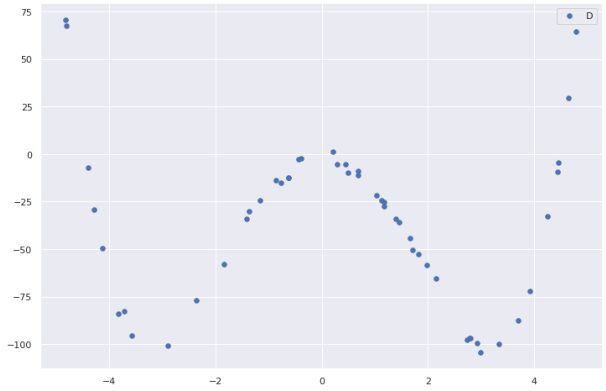


Figure 1: Your first dataset.

- ○ High bias and high variance
- ○ High bias and low variance
- ○ Low bias and high variance
- ○ Low bias and low variance

Alice's Arboretum

Alice wants to measure some oak trees that she planted over the years so that she can train a linear regression model for predicting tree height. She uses a small ruler to take measurements and notices that the measurements are much less reliable for taller trees. She wants to take this into consideration in her model.

a) `[3]` Why might she want to take this into consideration?

Alice decides to weigh each sample $(\boldsymbol{x}_i, y_i)$ by $k_i = 1/\sigma_i^2$, where σ_i^2 is her estimate of the variance for the measurement y_i. She gets the following weighted square loss function.

$$L(\hat{f}, D) = \sum_{i=1}^{|D|} k_i \left(y_i - \boldsymbol{w}^T \boldsymbol{x}_i\right)^2 = \|\mathbf{K}^{1/2}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w})\|^2,$$

where $\mathbf{K}$ is a diagonal matrix of the sample weights.
Alice has already collected some data and decides to plot her measurements in Figure 2. She has also added an **unweighted** least squares linear model to her plot.

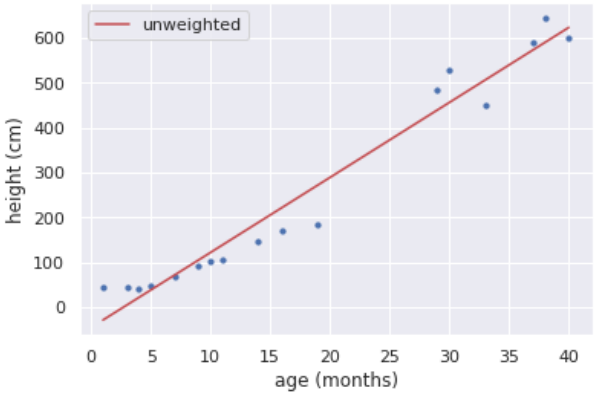


Figure 2: Alice's data and unweighted least squares linear model.

b) [3] How do you expect the weights of the weighted least squares model to differ?

c) [8] Show that the weights that minimize the weighted square loss function are given by:

$$\boldsymbol{w}^* = (\mathbf{X}^T\mathbf{K}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{K}\boldsymbol{y}$$

Solution

Multiple Choice

a) High bias and low variance is correct. The data is clearly higher degree than 3, probably coming from a degree 4 polynomial. So a polynomial of degree 3 will have a high bias, since it cannot fit the data closely, and a low variance.

Alice's Arboretum

a) She might not want the tall trees with high variance to have as strong an influence on the parameters of the model, so she might decide to give them a lower weight.

b) The fitted line will be flatter (lower gradient) [2 points], with a higher intercept term [not needed].

c) Similarly as in the lectures (Theorem 5.6) and exercises (Exercise 9.3), we can rewrite the loss function in matrix form as

$$\begin{aligned} L = \|\mathbf{K}^{1/2}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w})\|^2 &= (\boldsymbol{y} - \mathbf{X}\boldsymbol{w})^T \left(\mathbf{K}^{1/2}\right)^T \mathbf{K}^{1/2}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w}) \\ &= (\boldsymbol{y} - \mathbf{X}\boldsymbol{w})^T \mathbf{K}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w}) \end{aligned}$$

We can now differentiate with respect to $\boldsymbol{w}$ to find the optimal weights. Using the product rule and matrix differentiation rules (e.g., see Exercise Sheet 9), we get

$$\begin{aligned} &\frac{\partial L}{\partial \boldsymbol{w}} = -\left(\mathbf{X}^T\mathbf{K}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w})\right)^T - (\boldsymbol{y} - \mathbf{X}\boldsymbol{w})^T\mathbf{K}\mathbf{X} \stackrel{!}{=} \mathbf{0}^T \\ &\iff -\mathbf{X}^T\mathbf{K}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w}) - \mathbf{X}^T\mathbf{K}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w}) = \mathbf{0} \\ &\iff \mathbf{X}^T\mathbf{K}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w}) = \mathbf{0} \\ &\iff \mathbf{X}^T\mathbf{K}\boldsymbol{y} = \left(\mathbf{X}^T\mathbf{K}\mathbf{X}\right)\boldsymbol{w} \\ &\iff \boldsymbol{w} = \left(\mathbf{X}^T\mathbf{K}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{K}\boldsymbol{y}. \end{aligned}$$

6 Neural Networks (19 points)

Choose One

a) [3 points] Which one of the following is an advantage of Attention over RNNs?

- ○ 1. Attention can process inputs of any length.
- ○ 2. Attention can process the elements of an input sequence in parallel.
- ○ 3. Attention can find local patterns.
- ○ 4. None of the above.

Heart Beat

An electrocardiogram or ECG is a signal that describes the electric activity of the heart in terms of voltage over time. For each heart beat, ECG signals present a regular pattern, as shown in Figure 3, which can be written numerically as a one dimensional array $\boldsymbol{b} \in \mathbb{R}^n$:

$$\boldsymbol{b} = [0, ..., 0, 2, 4, 1, 0, ..., 0 - 3, 20, -6, 0, ..., 0, 2.5, 5, 0, ..., 0] + \boldsymbol{\epsilon}$$

where the number of 0 elements between peaks (i.e., between non-zero elements) can vary and $\boldsymbol{\epsilon} \in \mathbb{R}^n$ represents noise $\epsilon_i \sim [-1, 1]$.

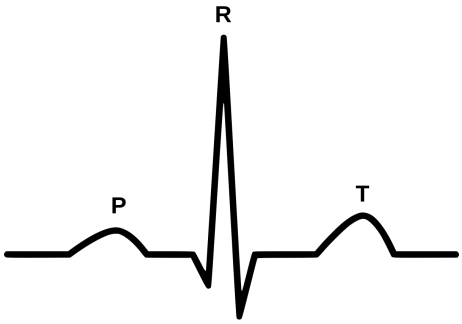


Figure 3: Beat pattern in an ECG. Source: Wikipedia

Several health-related applications require measuring the distance d between the P and T peaks. Here, we are going to build a two-layer neural network that outputs d. The input to the network is a vector $\boldsymbol{x} \in \mathbb{R}^n$ that contains one single beat with an arbitrary number of $0 + \epsilon$ elements before and after the beat. The first layer of our model is a convolutional layer that outputs a vector $\boldsymbol{h} \in \mathbb{R}^n$ with three 1's in exactly the positions of the P, R and T-peaks and 0 everywhere else.

a) [5] Why is a filter length of 1 not enough to detect the P, R and T-peaks?

b) [7] We use a filter $\boldsymbol{\phi}$ of length 2. We already know some of the values learned. Complete $\boldsymbol{\phi}$ as well as the intercept term β and the activation σ , where we use a piecewise activation function . (Remember, $\boldsymbol{h} = \sigma(\boldsymbol{\phi}(\boldsymbol{x}) + \beta)$)

$$\boldsymbol{\phi} = \begin{pmatrix} 10 & \square \end{pmatrix} \qquad \beta = \square \qquad \sigma(x) = \begin{cases} 1 & \square \\ 0 & \square \end{cases}$$

The second layer is a simple recurrent neural network that outputs a vector $\boldsymbol{y} \in \mathbb{R}^n$ with 1 in the elements between the P and T peaks (including one of them) and 0 everywhere else, such that $d = \sum_i^n y_i - 1$.

c) [4] Propose an output function $y_t = \sigma_2(h_t, s_t)$ and a state update function $s_{t+1} = g(s_t, h_t)$, with $s_0 = 0$. (Please suggest a σ_2 that is unusual in the context of neural networks.)

Solution

Multiple Choice

a) 2 is correct, while RNNs need to process all previous elements in a sequence to process the next one, the attention operation operates on each element individually, allowing for parallelization.

Heart Beat

a) The minimum filter length is 2. The R-peak and T-peaks could be detected with a filter of size 1, since their maximum value is larger than the values immediately to their right and left, including the noise. However, due to the noise a filter of size 1 would not be able to distinguish the exact location of the P peak, since the peak could take values between 3 and 5, intersecting with the range of the previous element, $[1, 3]$. However, a filter of size 2 could already distinguish the exact location of the P-peak from the previous element: the element immediately before would be $[[1, 3], [3, 5]]$ and the P-peak $[[3, 5], [0, 2]]$.

b) For the network to detect the exact peaks, the convolution operation should produce a larger value in the position of the peaks than in the other positions. Since the existence of noise makes that each point of the signal can take a value within a range, we need the minimum value in the peak positions to be larger than the maximum value of the other positions. This occurs only for $\phi_2 = -3$ as can be seen in the table below:

Event	**Pattern**	**min**	**max**
Flat	[[-1,1],[-1,1]]	13	-13
P-peak	[[-1,1],[-1,3]]	-19	7
P-peak	[[1,3],[3,5]]	-5	21
P-peak	**[[3,5],[0,2]]**	**24**	**50**
P-peak	[[0,2],[-1,1]]	-3	23
R-peak	[[-1,1],[-4,-2]]	1	22
R-peak	[[-4,-2],[19,21]]	-103	-77
R-peak	**[[19,21],[-7,-5]]**	**175**	**189**
R-peak	[[-7,-5],[-1,1]]	-73	-53
T-peak	[[-1,1],[1.5,3.5]]	0.5	5.5
T-peak	[[1.5,3.5],[4,6]]	-3	23
T-peak	**[[4,6],[-1,1]]**	**37**	**63**

Table 1: Convolution results for $\phi_2 = -3$, where the ranges in the pattern column represent the minimum and maximum value that each element of the array $\mathbf{b}$ can take. The rows correspond to all the possible positions of the convolutional filter $\boldsymbol{\phi}$

Filling a similar table with other (integer) values for ϕ_2 shows that the different patterns would overlap.

From these results, we see that if we choose $\beta = -23$, the only values larger than 0 correspond to the position of the peaks. Choosing the activation function to be the indicator function:

$$\sigma(x) = \begin{cases} 1 & \text{if x} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

We obtain the desired output, i.e., 1 in the peaks, 0 everywhere else.

c) We need the recurrent network to output a 1 after it sees the first 1 and until it finds the third 1. This way, we can simply choose:

$$y_t = \sigma_2(s_t)$$

$$s_{t+1} = s_t + h_t$$

and

$$\sigma_2(x) = \begin{cases} 1 & \text{if x} > 0, \text{if x} < 3 \\ 0 & \text{otherwise.} \end{cases}$$

7 Computability (18 points)

Choose One

a) `[3 points]` Consider the following modified variant of the halting problem.

`Any-halting problem`: *Given a program P, does there exist an input x such that $P(x)$ halts on x?*

How hard is this problem?

- ○ Solvable in polynomial time.
- ○ Decidable but NP-hard.
- ○ Undecidable.
- ○ None of the above.

b) `[3 points]` Consider a modified variant of the PCP problem where each domino has three words written on it (top, middle, bottom), and we need to find a sequence of dominos such that the concatenation of words is the same on all three levels. How hard is this variant of the PCP problem?

- ○ Solvable in polynomial time.
- ○ Decidable but NP-hard.
- ○ Undecidable.
- ○ None of the above.

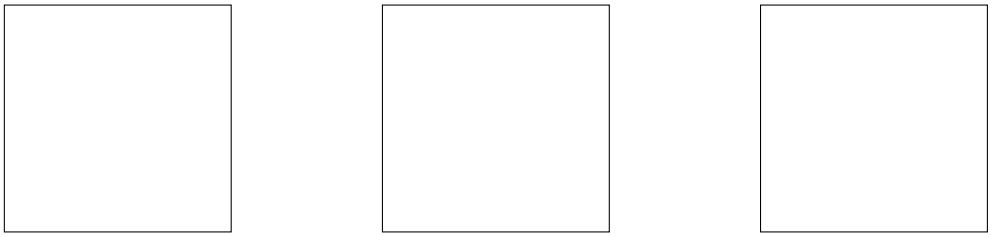
Tilings

In this task, you are given a specific number of tiles, and you have to assign colors to each side of the tiles such that they provide a valid tiling of the plane, and also fulfill specific properties. The following rules apply to each subexercise:

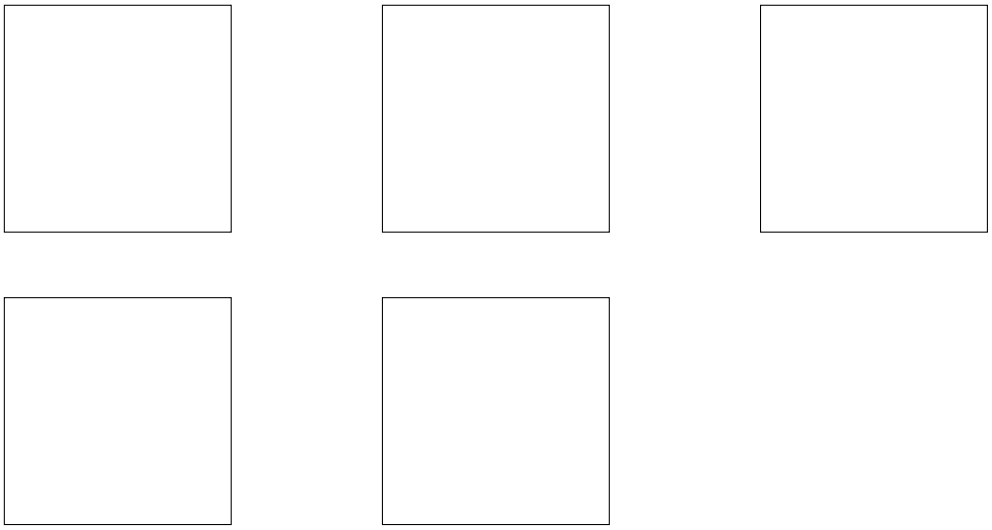
- you have to use all the provided tiles for your tileset,
- you can use any number of colors,
- each tile has to be different, i.e. each pair of tiles must differ in at least one of the four sides.

Design a tileset such that

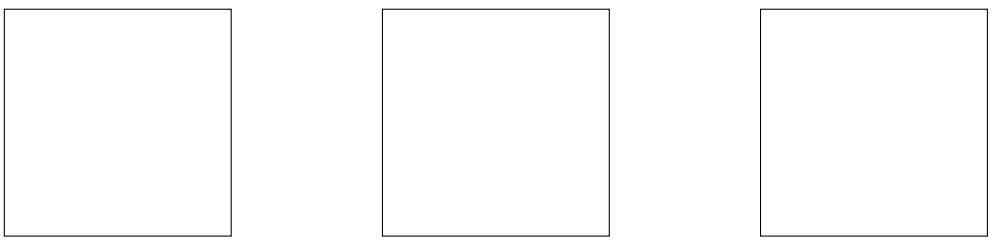
a) `[4 points]` any valid tiling uses all three tiles.



b) `[4 points]` in any valid tiling of the plane, it holds that (i) all five tiles are used, and (ii) there is no row or column that only contains a single kind of tile.



c) `[4 points]` there exists a valid tiling of the plane such that the first tile is used exactly once.



Solution

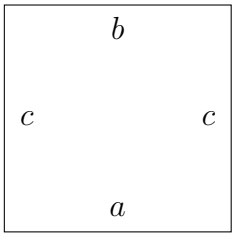
Multiple Choice

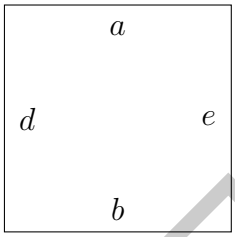
a) *Undecidable* – we can apply almost the same reduction as with the mortality problem (Algorithm 7.10), but instead of terminating in line 5, we now go into an infinite loop instead.

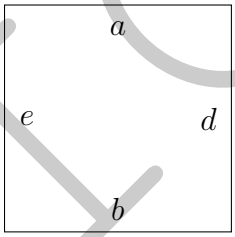
b) *Undecidable* – for a reduction, we can create a new set where the middle word is the same as the bottom one in each domino; in this case, the new domino set provides a valid sequence if and only if the original set does.

Tilings

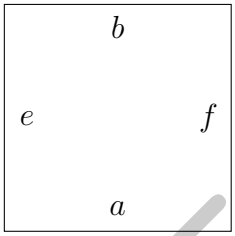
a) Solution:

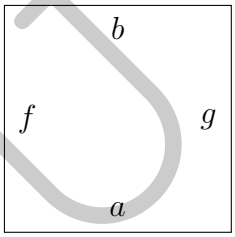


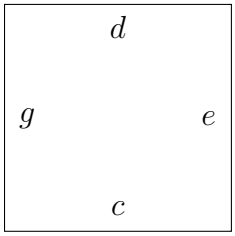


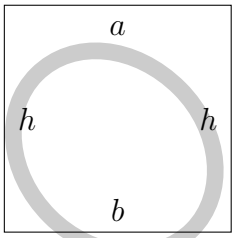


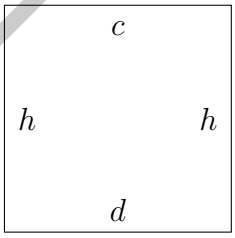
b) Solution:



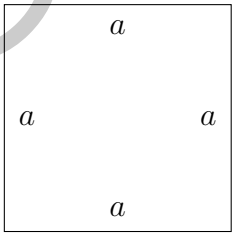


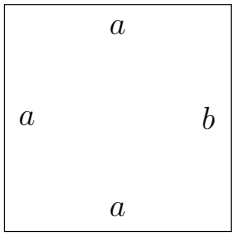


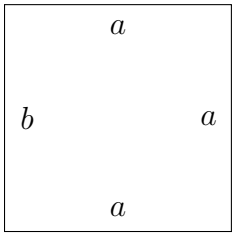




c) Solution:







Replacements

Here, you can find replacement templates for the fill in tasks, in case you messed up. If you use these, please **clearly** indicate which one we should consider.

```python
# Assume input n to be a non-negative integer.
def fibo2(n, memo = None):
    if not memo: memo = {0: 1, 1: 1}
    ##### Fill below #####


    ######################
    answer = fibo2(n-1, memo) + fibo2(n-2, memo)
    ##### Fill below #####


    ######################
    return answer
```

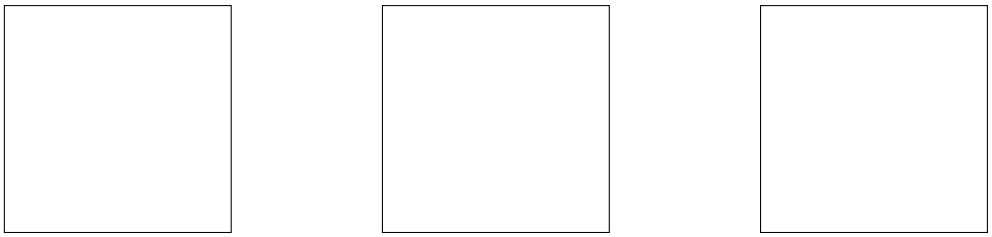
Protocol 2

Alice	**Bob**
has (k_p^A, k_s^A), knows k_p^B	has (k_p^B, k_s^B), knows k_p^A

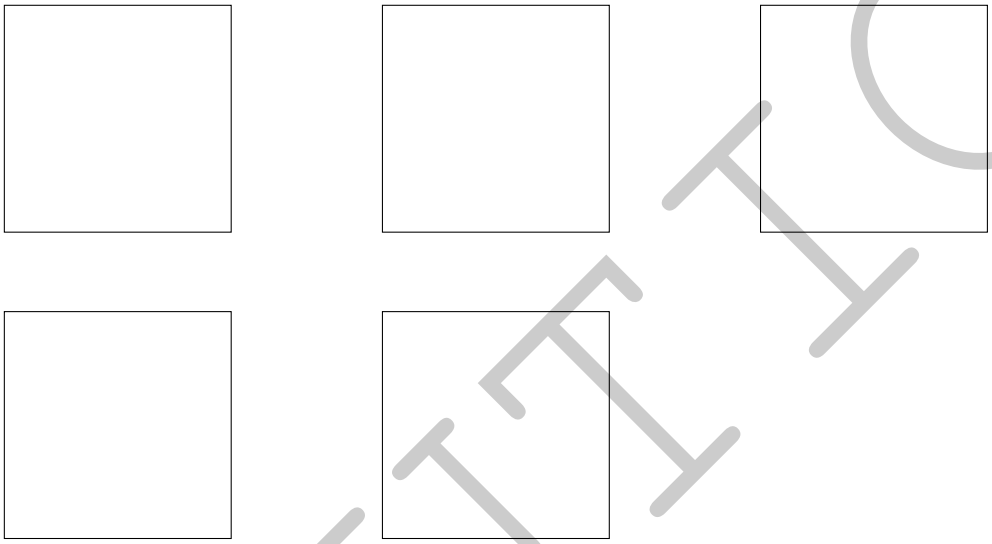
$\longrightarrow$

$\phi = \begin{pmatrix} 10 & \square \end{pmatrix}$ $\quad \beta = \square$ $\quad \sigma(x) = \begin{cases} 1 & \square \\ 0 & \end{cases}$

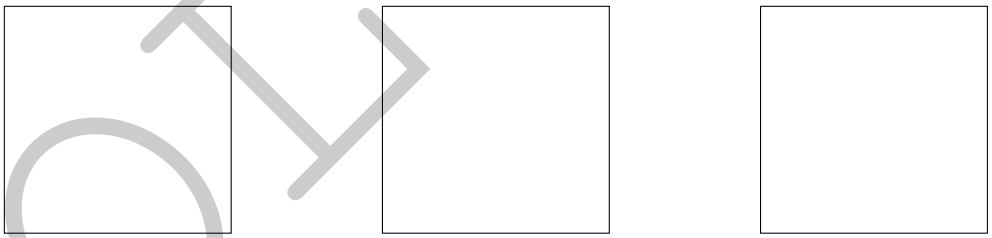
a)



b)



c)



This page is intentionally blank