Chair of Application and Middleware Systems
Department of Informatics
Technical University of Munich

Personal sticker

S5030

**Compliance to the code of conduct**
I hereby assure that I solve and submit this exam myself under my own name by only using the allowed tools listed below.

Signature or full name if no pen input available

# Distributed Systems

| | | | |
|---|---|---|---|
| **Exam:** | IN2259 / DS Final | **Date:** | Thursday 25th February, 2021 |
| **Examiner:** | Prof. Dr. Hans-Arno Jacobsen | **Time:** | 16:30 – 17:45 |

## Working instructions

- This exam consists of **14 pages** with a total of **8 problems**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 100 credits.
- Detaching pages from the exam is prohibited.
- Allowed resources:
  - The exam is an open book, and course materials are allowed to be used.
  - **Any collaboration is cheating and will be reported to administrations!**
- Unless stated differently, please use the algorithms that have been introduced in the lecture or the tutorial.
  Self-defined algorithms may reduce the awarded credits.
- **Do not write with red or green colors.**

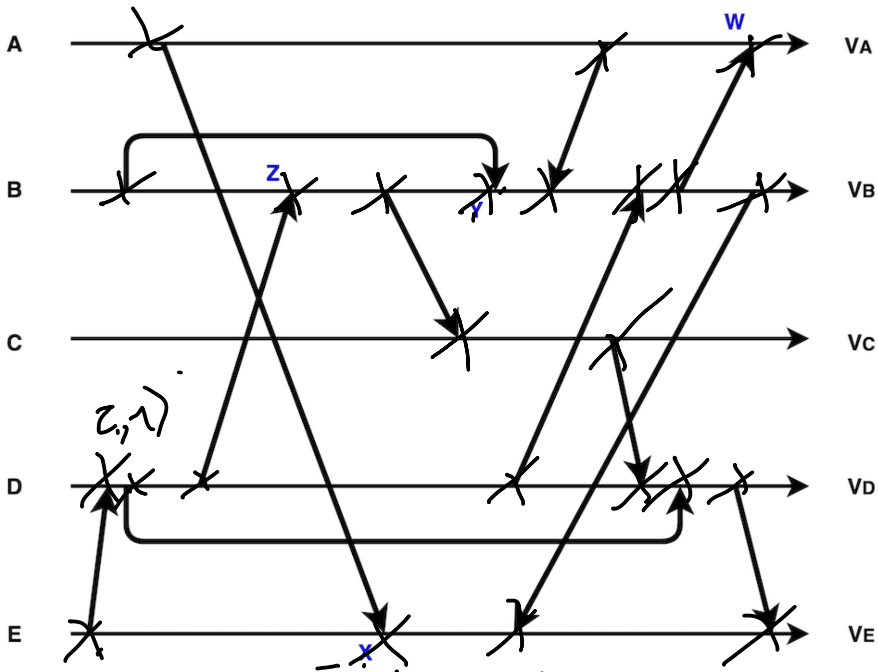
Left room from ____________ to ____________ / Early submission at ____________

☐ Exam empty

## Problem 1 Logical Clocks (19.5 credits)

Consider a distributed system consisting of five processes. These processes broadcast messages among themselves and make use of logical clocks of $\{V_A, V_B, V_C, V_D, C_E\}$ for timestamping the messages.

a) For each process, determine the value of the logical clock at the **end** of the time/event diagram according to the **Vector Clock** algorithm. Only write the final value of clocks in the table below. All clocks **must** be initialised with zero.




Z=[0,1,0,3,1] Y=[0,3,0,3,1]

| | 1st Element | 2nd Element | 3rd Element | 4th Element | 5th Element |
|---|---|---|---|---|---|
| $V_A$ | ~~3~~ ~~1~~ 2 | 7 | 0 | 4 | 1 |
| $V_B$ | 2 | 8 | 0 | ~~3~~ 4 | 1 |
| $V_C$ | 0 | 3 | ~~1~~ 2 | 3 | 1 |
| $V_D$ | 0 | 3 | 2 | 7 ~~6~~ | 1 |
| $V_E$ | 2 ~~1~~ | 8 | 2 | ~~6~~ 7 | 4 |

b) Using the results of the previous task, **show and prove** whether the happened-before relationship (->) between the following event pairs holds or events are concurrent (||). The events are marked on the time/event diagram.

**1.** (X, Y) X = [1,0,0,0,2] ~~Y~~ = W = [VA]

X || Y ; None of the clocks ist larger or smaller at every entry

**2.** (Y, Z)

Z -> Y : [0,1,0,3,1] < [0,3,0,3,1]

**3.** (X, W)

X -> ~~W~~ W : [1,0,0,0,2] < [2,8,2,7,4

c) Consider the following distributed algorithm to implement causal-ordered (CO) broadcast. $L_i$ denotes the Lamport clock value at process $P_i$. *B-Broadcast* and *B-Deliver* denote the corresponding two basic primitives. Read the algorithm and answer the questions below.

0
1
2
3
4

**CO-Broadcast at Process $P_i$:**
*On Initialisation:*
$L_i := 0$
*To CO-Broadcast message M:*
$L_i := L_i + 1$
*B-Broadcast*($\langle L_i, M\rangle$)
*On B-Deliver*($\langle L_j, M\rangle$) *from* $P_j$ ($j \neq i$):
Place $\langle L_j, M\rangle$ in hold-back queue
Wait until $L_j = L_i + 1$
Remove message from queue and *CO-Deliver M*
$L_i := L_i + 1$

**1.** This algorithm cannot satisfy the requirements of the causal-ordered broadcast. What is the reason for that?

~~We have to use the La~~
Lamport clock does not guarantee causal consist.

**2.** Give a hint for fixing this algorithm so that the requirements of the causal-ordered broadcast are addressed.

we have to use the Vector clock

## Problem 2 Leader Election / Time Adjustment (12 credits)

Consider a distributed system consisting of eight processes of $\{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8\}$. Answer the following questions.

0 1 2 3 4

a) Process $P_1$ and $P_3$ suspect the demise of network's leader. Both processes simultaneously initialize a leader election based on **Bully Algorithm**. Explain the procedure for electing the leader and determine which process will be elected to be the network's new leader.

Both will send out election messages. Processes with higher IDs will answer and will start an election instead. Eventually P8 starts an election and nobody will reply. After a timeout, it sends a coordination message because it won

0 1 2 3 4

b) The newly elected leader performs a clock synchronization round using the **Berkeley Algorithm**. The threshold value for acceptable deviation is $\delta = \pm 2000ms$. Show what times are selected for calculating $T_{ref}$ and what reference time $T_{ref}$ does the leader calculate for synchronization?

| Process | Local Time (hh:mm:ss:fff) |
|---|---|
| $P_1$ | 11:10:41:000 |
| $P_2$ | 11:10:48:000 |
| $P_3$ | 11:10:40:000 |
| $P_4$ | 11:10:39:000 |
| $P_5$ | 11:10:47:000 |
| $P_6$ | 11:10:37:000 |
| $P_7$ | 11:10:34:000 |
| $P_8$ | 11:10:40:000 |

we use the times of $P_1, P_3, P_4, P_8$

$T_{ref} = 11:10:40:0000$

c) Based on the calculated $T_{ref}$ time, what values does the leader calculate and send to other processes? Fill in the table below.

(in seconds)

| Process | Value |
|---|---|
| $P_1$ | -1 |
| $P_2$ | -8 |
| $P_3$ | 0 |
| $P_4$ | +1 |
| $P_5$ | -7 |
| $P_6$ | +3 |
| $P_7$ | +6 |
| $P_8$ | 0 |

0 1 2 3 4

Page empty

## Problem 3 MapReduce (8 credits)

Formulate the below algorithm for MapReduce. Explain how the input is mapped into **(key, value)** pairs, i.e., specify what is the key and what is the associated value in each pair, and, if needed, how the key(s) and value(s) are computed. Then, explain how the **(key, value)** pairs produced by the map stage are processed by the reduce stage to get the final result.

0 1 2 3 4 5 6 7 8

Consider a set of documents (each with a documentID and a documentContent), where the content consists of a list of words. Provide a MapReduce program that generates a list of locations (word position in the documentID) for each word occurrence:

word,

**File Content:**

$$\text{Document1} \; : \; \{\text{Word1 Word2 Word1 Word3}\}$$
$$\underbrace{\text{Document2}}_{documentId} \; : \; \underbrace{\{\text{Word4 Word2 Word4 Word2 Word1}\}}_{documentContent}$$

ID

doc ID cont

**Final Output:**

Word1: {Document1 : [1, 3], Document2 : [5]}
Word2: {Document1 : [2], Document2 : [2, 4]}
Word3: {Document1 : [4]}
Word4: {Document2 : [1, 3]}

Document
(~~ID~~, word

**1. Map:** What does the map function do?

it lists the indices of all words in one document

**2. Map Input:** Define the Map input in the **(key, value)** format.

(documentId, documentContent)

**3. Map Output:** Define the Map output in the **(key, value)** format.

(word, (~~D~~ doc.Id, list(index))

**4. Reduce:** What does the reduce function do?

for ~~a~~ specific word, it appends the indices of the documents from the reduce phase

**5. Reduce Input:** Define the Reduce input in the **(key, value)** format.

(word, list(docId, list(index)))

**6. Reduce Output:** Define the Reduce output in the **(key, value)** format.

list(word(list(docId, list(index))))

## Problem 4 Consistency Models (18 credits)

a) Consider the execution depicted in Figure 4.1. Is it **linearizable** (Yes/No)? Is it **sequentially consistent** (Yes/No)? For both questions, if yes, valid execution sequences **must** be written. If no, explain. (**Hint:** a Yes without providing the sequence or a No without explanation, is zero point for every part.)

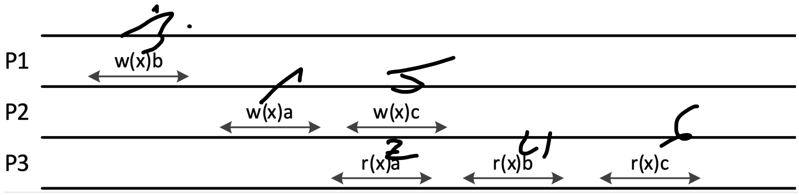



Figure 4.1: Execution.

not linearizable, but sequentially consistent
-> w(x)a r(x)a w(x)b r(x)b w(x)c r(x)c
it is not linear. because the notion of
a global time is violated

b) Consider the execution depicted in Figure 4.2. Is it **lineralizable** (Yes/No)? Assuming each operation is performed at the start of its time interval, is it **strictly consistent** (Yes/No)? For both questions, if yes, valid execution sequences **must** be written. If no, explain.

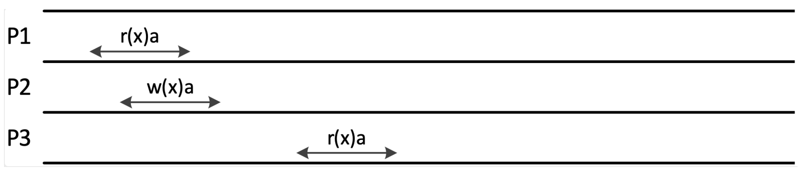



Figure 4.2: Execution.

It is linearizable: w(x)a r(x)a r(x)a.
It is not strictly consist. bc
r(x)a would be read before a
was written (w(x)a)

c) Consider the execution depicted in Figure 4.3. Is the execution **sequentially consistent** (Yes/No)? Is it **causally consistent** (Yes/No)? For both questions, if yes, valid execution sequences **must** be written. If no, explain.

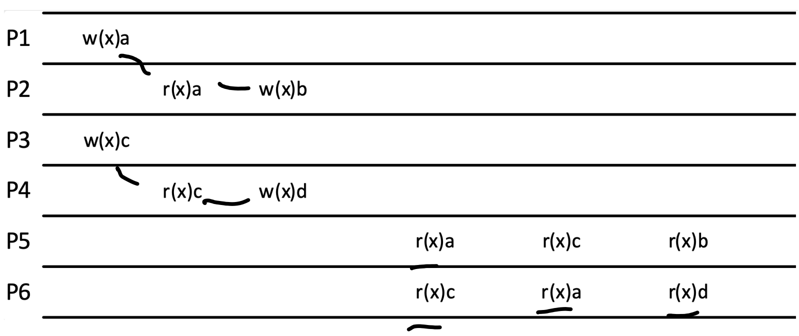



Figure 4.3: Execution.

It is not sequent. consist: in P5 we have r(x)a r(x)c ∧ r(x)c r(x)a in P6
-> no unique order
P5: w(x)a r(x)a w(x)c r(x)c w(x)d r(x)d
~~P6: w(x)a r(x)a~~

0 1 2 3

d) Is the execution depicted in Figure 4.4 **sequentially consistent** (Yes/No)? Is it **causally consistent** (Yes/No)? For both questions, if yes, valid execution sequences **must** be written. If no, explain.

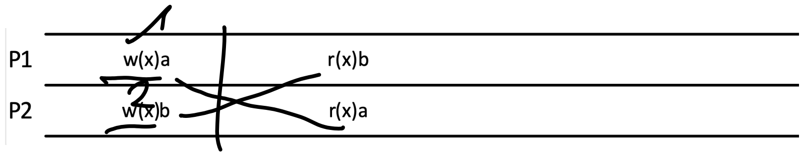



Figure 4.4: Execution.

It is not sequent. consist but ~~it~~

0 1 2

e) Is the execution depicted in Figure 4.5 **FIFO consistent** (Yes/No)? Is it **causally consistent** (Yes/No)? For both questions, if yes, valid execution sequences **must** be written. If no, explain.

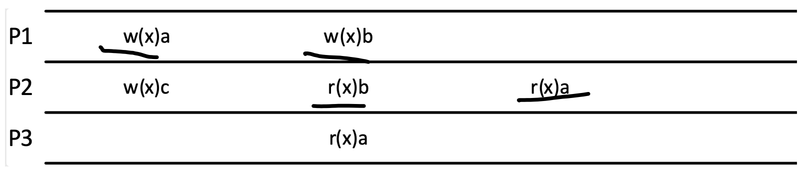
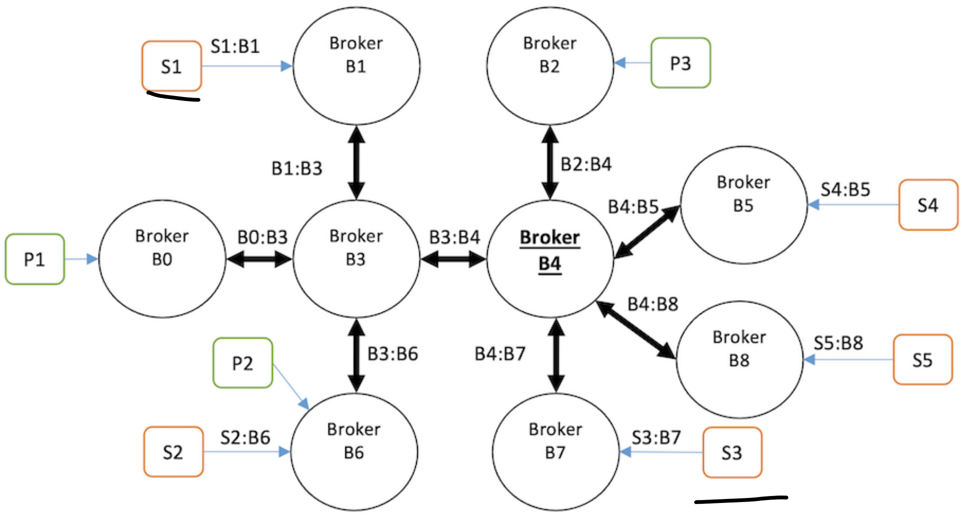



Figure 4.5: Execution.

It is not FIFO consistent:
r(x)b read before r(x)a
=> thus also not causally consist.

## Problem 5 Publish/Subscribe (10.5 credits)

Consider an overlay broker network based on a **rendezvous-based routing model** containing nine brokers, as demonstrated in the figure below. Consider all subscribers are already subscribers and stored in the **Rendezvous Broker B4** which uses Bloom filters. Also, each link has a unique ID. Answer the following questions.




a) Publisher **P3** creates a publication that matches the subscriptions of **S1** and **S3**. Specify which links are added to the bloom filter.

0 1 2

P3:B2, B2:B4, B4:B7, B7:S3
P3:B2, B4:B3, B3:B1, B1:S1

b) The rendezvous broker creates a bloom filter of size 16 for routing the matching publications. The table below contains the hashed value of each link using four hash functions. The bloom filter uses all four hash functions in the order when inserting and looking up values.
Publisher **P3** creates a publication that matches the subscription of **S1**. The rendezvous broker adds the links required for delivering the publication to the bloom filter below. Show the effect of adding the links to the bloom filter by filling in the vector below.

0 1 2 3 4 5 6 7 8

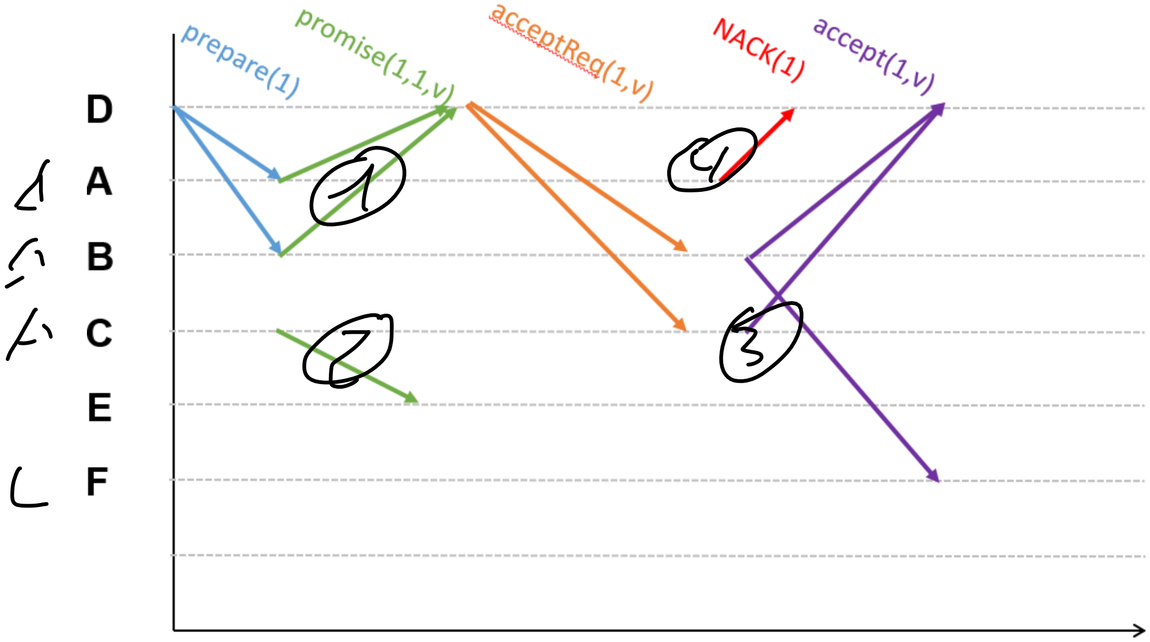
| Edge | $H_1$ | $H_2$ | $H_3$ | $H_4$ |
|---|---|---|---|---|
| B0:B3 | 14 | 15 | 1 | 10 |
| B1:B3 | 9 | 1 | 7 | 12 |
| B2:B4 | 10 | 2 | 11 | 16 |
| B3:B4 | 8 | 14 | 16 | 7 |
| B3:B6 | 6 | 15 | 9 | 10 |
| B4:B5 | 2 | 15 | 3 | 14 |
| B4:B7 | 4 | 13 | 2 | 12 |
| B4:B8 | 2 | 7 | 15 | 5 |
| S1:B1 | 16 | 11 | 8 | 2 |
| S2:B6 | 9 | 7 | 13 | 15 |
| S3:B7 | 10 | 4 | 9 | 2 |
| S4:B5 | 4 | 7 | 6 | 15 |
| S5:B8 | 15 | 7 | 14 | 2 |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

## Problem 6 Paxos (6 credits)

0 1 2 3 4 5 6

The figure below shows the execution of Basic Paxos with two proposers (D, E), three acceptors (A, B, C), and one learner (F). This diagram contains precisely four mistakes that are not allowed in Basic Paxos. Identify them and write them below.




(1) Acceptors have not accepted value before thus: promise(1,-,-)

(2) C has to answer to D if it had received a proposal

(3) C has to send the accept msg. also to the learner F

(4) A cannot reply with a NACK here because it did not receive the acceptReq

## Problem 7 Peer-To-Peer Systems (16 credits)

a) Consider the Pastry network, where every node ID $\in \{0, 1, 2, 3, 4\}^5$ and is represented by a 4-digit string. The network contains the following nodes:

{ 0014, 0100, 0103, 0202, 0213, 0231, 0322, 0431, 1041, 1111, 1243, 1341, 1432, 2030, 2230, 2323, 2402, 2413, 3010, 3113, 3133, 3240, 3302, 3343, 3424, 4004, 4030, 4133, 4201, 4331, 4341, 4414 }

Complete the routing table of node **2323** and **3010** in the table below: 6

Table 7.1: Routing Table **2323**

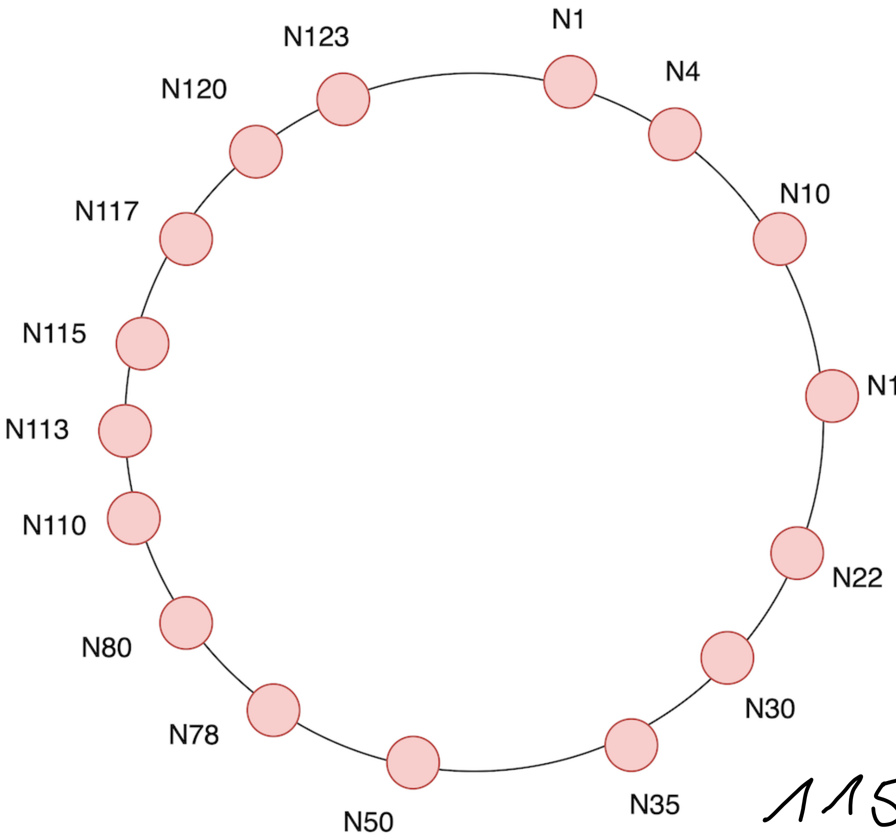
| RT | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0014 | 1041 | 2323 | 3010 | 4004 |
| 1 | 2030 | – | 2230 | 2323 | 2402 |
| 2 | 23– | – | 2323 | – | – |
| 3 | – | – | – | 2323 | – |

Table 7.2: Routing Table **3010**

| RT | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0014 | 1041 | 2323 | 3010 | 4004 |
| 1 | 3010 | 3113 | 3240 | 3302 | 3424 |
| 2 | – | 3010 | – | – | – |
| 3 | 3010 | – | – | – | – |

0 1 2 3

b) Let $C$ be the Chord ring given in the figure below. The network contains several peers, each of which was mapped to a node $N_i \in C$ by a uniform hash function with bit-string length $m = 7$. Consider the standard Chord implementation where peer nodes store a finger table. Fill in the finger table of the peer at node **N115**:




1151
2
4
8
16
32
64

Table 7.3: Finger Table **N115**

| Peer ID | Successor |
|---|---|
| 116 | N117 |
| 117 | N117 |
| 119 | N120 |
| 123 | N123 |
| 5 | N10 |
| 19 | N22 |
| 51 | N78 |

## Problem 8 Short Answers (10 credits)

a) Consider a distributed system that uses gossiping for lazy replication. According to the CAP theorem, what is this system called?

AP

b) Consider a Byzantine general's problem with five faulty processes. What is the minimum number of processes to reach the consensus based on Lamport's original solution?

$3f + 1 \rightarrow 16$

c) Given 10 data pieces and 10 parity pieces, how many failures does a Reed-Solomon encoding tolerate?

10

d) Conflict-Free Replicated Data Sets are used to provide what type of consistency?

~~Strong~~ Eventual Consistency

e) Which theoretical property does Paxos NOT guarantee?

liveness

f) Consider a cryptocurrency that operates like Bitcoin, and the reward for mining a block is 50 coins, and each block contains 200 transactions with an average fee of 0.5 coins for each transaction. What is the total gain for mining a block?

$50 + 0.5 \times 200 = 150$

**Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.**