OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Magdeburg, den 13.09.2024

Re-Examination for the Lecture
Introduction to Computer Science for Engineers (Winter 23/24)

Nachname: _____

Vorname: _____

Matrikelnummer: _____

Please consider the following advices:

- You have **120 minutes** to complete the exam.
- Please place your passport and student ID card in front of your table.
- The exam consists of 14 pages. Please check if you received all of them.
- Before working on a task, read the task description thoroughly. Put your answers into the corresponding boxes or mark your solutions directly within the task description (where appropriate). Should your solution not fit the given space, use the backside of the **previous** page to continue writing. Do not use your own paper.
- Strike out wrong solutions.
- Do not use pencils or red or green ink.
- No supplementary material is allowed except a pen.
- Should you hand in more than one solution to a task, the task will not be graded at all.
- Should you have any question during the exam, please raise your hand so we can come to you and help.
- Leaving the room has to be protocolled and is only allowed for one person at a time. When wanting to leave, raise your hand so we can tell you when you are allowed to leave.
- **Cheating attempts lead to you failing the exam**.

Good Luck! 😀

Aufgabe	1	2	3	4	5	6	7	8	$\sum$
maximale Punkte	11	4	6	8	10	9	7	5	60
erreichte Punkte									

Magdeburg, den 13.09.2024

Aufgabe 1 *: Fundamental Algorithms*

a *) The following code implements a recursive version of the binary search algorithm.*

```
1  def find(a: list[int], x: int) -> int | None:
2      def _find(a: list[int], x: int, l: int, r: int) -> int | None:
3          if l > r:
4              return None
5          m: int = (l+r) // 2
6          if a[m] == x:
7              return m
8
9          if x < a[m]:
10             return _find(a, x, l, m - 1)
11         else:
12             return _find(a, x, m + 1, r)
13
14     return _find(a, x, 0, len(a)-1)
```

- *What is the meaning of the return value of* `find`*?*
- *Define and implement an iterative function* `ifind` *that implements binary search without using recursion. Do not use any libraries or built-in functions that would solve the problem directly.*
- *Assume that you are allowed to make only 11 comparisons. What is the size of the largest array that you can search and guarantee that you return the correct answer?*
  *(**Hint:** To make things a bit easier, you should only consider the comparison in line 9 as* comparison *in the context of this question.)*

[5 points]

Magdeburg, den 13.09.2024

b *) Given the following function, explain what each function calculates and what its* worst case *runtime complexity is (Big-O notation).*

```
1  def a(n: int) -> int:
2      if n <= 1:
3          return 1
4      else:
5          return n * a(n-1)
6
7
8  def b(n: int) -> int:
9      if n <= 0:
10         return 0
11     else:
12         return 1 + b(n//2)
13
14 def c(n: int) -> int:
15     if n <= 0:
16         return 0
17     elif n <= 2:
18         return 1
19     else:
20         return c(n-2) + c(n-1)
```

[6 points]

/ 11

Magdeburg, den 13.09.2024

Aufgabe 2 : *Object Oriented Programming*

a) *Consider the following class hierarchy, which output is generated by the lines 30, 33, 34, and 37?*

```
1  class Pokemon:
2      def attack(self, s: str) -> None:
3          raise NotImplementedError
4
5      def move(self, target: str) -> None:
6          print(f"Teleporting to {target}.")
7
8  class Eevee(Pokemon):
9      def attack(self, s: str = "") -> None:
10         print("Tackle!")
11
12     def move(self, target: str) -> None:
13         print(f"Run away from {target}.")
14
15 class Vaporeon(Eevee):
16     def attack(self, target: str) -> None:
17         super().move(target)
18         print("Water Gun!")
19         self.attack(target)
20
21 class Magikarp(Pokemon):
22     def attack(self, s: str) -> None:
23         super().attack(s)
24
25     def move(self, target: str) -> None:
26         print(f"Flop around.")
27
28 if __name__ == "__main__":
29     garados: Magikarp = Magikarp()
30     garados.move("Kanto")
31
32     jolteon: Eevee = Eevee()
33     jolteon.attack()
34     jolteon.move("Alola")
35
36     flareon: Pokemon = Vaporeon()
37     flareon.attack("Pikachu")
```

	Output
line 30:	
line 33:	
line 34:	
line 37:	

[4 points]

/ 4

Magdeburg, den 13.09.2024

Aufgabe 3 *: Lists*

a *)*

```
1  class DoubleLinkedNode:
2      def __init__(self, item=None, prev=None, next=None):
3          self._item = item
4          self._prev = prev
5          self._next = next
6
7
8  class DoubleLinkedList:
9      def __init__(self):
10         self._head = DoubleLinkedNode()                    # an empty DLL always
11         self._tail = DoubleLinkedNode(prev=self._head)     # contains at least
12         self._head._next = self._tail                      # two empty nodes
13
14     def insert_before(self, node: DoubleLinkedNode):
15         pass
16
17     def remove(self, node: DoubleLinkedNode):
18         pass
```

Implement the two missing functions `insert_before` *and* `remove`*. You may assume that all argumentes are sufficiently instantiated (i.e., not* `None`*) and – in the case of* `remove` *– are contained in the double linked list. You can also directly access all members of the* `DoubleLinkedNode` *class without using getter or setter functions. [4 points]*

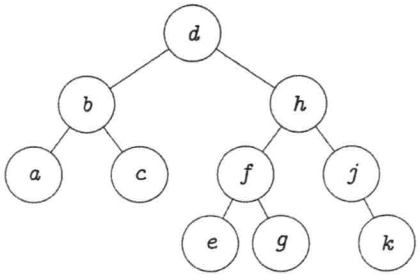
Magdeburg, den 13.09.2024

b *) The implementation of the doubly linked list from the lecture slides and the previous task use seperate dummy nodes for the* `_head` *and* `_tail` *pointers. Why? What would change, if we just referenced the first and last element directly? [2 points]*

Magdeburg, den 13.09.2024

Aufgabe 4 *: Tree Structures*

a *) Consider the following binary search tree:*



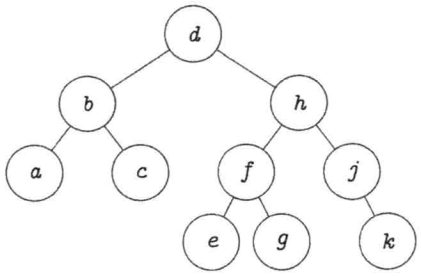
- *Delete – in that order – the nodes* `a`*,* `j`*, and* `d`*. Draw the resulting tree after every removal.*

[4 points]

Magdeburg, den 13.09.2024

b *) Given the following binary tree, what is the output of*

- *a preorder traversal,*
- *an inorder traversal,*
- *a postorder traversal,*
- *a levelorder traversal?*



preorder traversal:	
inorder traversal:	
postorder traversal:	
levelorder traversal:	

[4 points]

/ 8

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Magdeburg, den 13.09.2024

Aufgabe 5 *: Balanced Trees*

a *)*

- *Explain* ***briefly*** *what an AVL tree is.*
- *Insert the following sequence of numbers into an AVL tree. Sketch the tree after every insertion and before/after every reorganisation step.*

`[5, 2, 1, 6, 3, 4, 7, 0]`

[5 points]

Magdeburg, den 13.09.2024

b)

- *Explain **briefly** what an 2-3-4 tree is.*
- *Insert the following sequence of numbers into an 2-3-4 tree. Sketch the tree after every insertion and before/after every reorganisation step.*

`[50, 20, 60, 10, 8, 15, 32, 26, 48, 42]`

[5 points]

/ 10

Magdeburg, den 13.09.2024

Aufgabe 6 : *Hashs*

a) *Insert the following sequence of numbers into a hash table* $\mathcal{H}$.

```
[15, 9, 1, 10, 8, 3]
```

The hash table has the size $m = 7$. *Consider the following two hash functions:*

$$h_1(x) = 2x - 1$$

$$h_2(x) = 3x + 1.$$

x	1	3	8	9	10	15
$h_1(x)$						
$h_2(x)$						

---

1. *Use open addressing with linear probing for collission handling. Use* h_1 *as hash function! Additionally count the number of collissions.*

i	0	1	2	3	4	5	6
$\mathcal{H}[i]$							

2. *Use open addressing with double hashing for collission handling. Use* h_1 *and* h_2 *as hash functions! Additionally count the number of collissions.*

i	0	1	2	3	4	5	6
$\mathcal{H}[i]$							

3. *What advantage does double hashing have in general over linear probing?*
4. *To avoid having too many collissions we might consider a randomized hash function. Why is this a bad idea?*

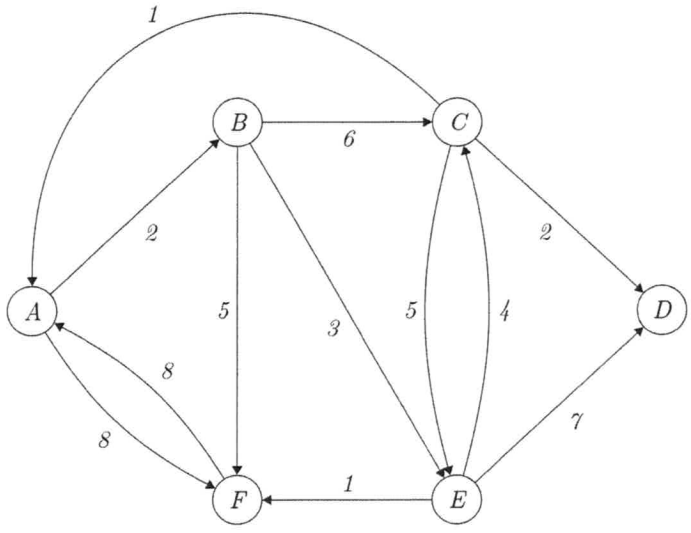
[9 points]

/ 9

Magdeburg, den 13.09.2024

Aufgabe 7 *: Graphs*

a *) Apply Dijkstra's Algorithm on the following graph, starting from node* `A`*.*

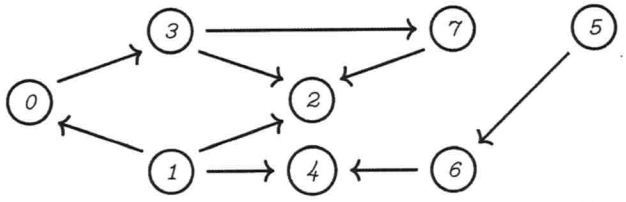


	A	*B*	*C*	*D*	*E*	*F*
parent	-					
dist. to A	*0*					

[4 points]

Magdeburg, den 13.09.2024

b *) You are given the following graph:*



Calculate a topological ordering of this graph. If at any stage more than one node is available, choose the one with the lower number.
[3 points]

/ 7

Magdeburg, den 13.09.2024

Aufgabe 8 *: True or False*

a *) Decide whether the following statements are true or false. You do not need to justify your answer. Every correct answer gives 0.5 points, every wrong answer gives -0.5 points. Not answering a question gives neither a malus nor a bonus. You will score at least 0 points on this task.*

1. $log_3 2^n \in \mathcal{O}(n)$
2. *Bubble sort can be faster than insertion sort in the best case.*
3. *The fastest way to build a heap from an array is in* $\mathcal{O}(n)$.
4. *The* B *in B-Tree stands for* binary.
5. *Merge sort can be used as an external sorting algorithm.*
6. *Python programs are compiled before they are executed.*
7. *In Python classes can inherit from more than one parent class.*
8. *Prefixing a class member with* __ *means that it cannot be accessed from outside of the defining class.*
9. *In a binary min heap the largest element can be found in one of the leaves.*
10. *Solving the knapsack problem with dynamic programming requires* $\mathcal{O}(n^C)$ *steps.*

	true	*false*
1	☐	☐
2	☐	☐
3	☐	☐
4	☐	☐
5	☐	☐
6	☐	☐
7	☐	☐
8	☐	☐
9	☐	☐
10	☐	☐

[5 points]

/ 5