Online Exam Instructions

General Instructions

- The exam is open book. You may use the lecture notes, the slides, the exercises and solutions, your own submissions to lab exercises, your own notes and a C compiler on your own computer.
- You may *not* use any other sources and you may not communicate with anyone else during the exam.
- Turn off or silence your phone, as you would during a regular exam.
- Make sure that you have a stable internet connection and a quite environment.
- Quit all messaging or email applications on your computer and disable all notifications.
- If you want to ask a question during the exam, please join the exam session on Blackboard Collaborate and *send your question via chat only to the moderators.* Do not use the "Everybody" chat.
- Note: points are only an indication. The weight of each exercise might still change.

Exercise Types

- For exercises marked with **TEXT** you must enter the answer directly in the text field on Nestor.
- For exercises marked with **PDF** you must upload a PDF file containing a hand-written answer, photographed or scanned.
- For exercises marked with **CODE** you must upload a C file with source code. This must be a single file which only contains the function asked for and any helper functions you wrote yourself. These files will be manually graded and do not have to compile.

Nestor Instructions

1. Click on the title of a question to open it.
2. Your text is saved automatically every two minutes. If you want to leave the question temporarily, click on 'Save as Draft'/'Als concept opslaan'. You will return to the main page of the exam.
3. When you have completed a question and wish to submit it, click on 'Save and Finish'/'Opslaan en inleveren'. Note: after this you will no longer be able to edit your answer!
4. You can also finish all your questions at once. Go to the main page and click on 'Finish all essays'/'Alle essays inleveren'. After this, you will no longer be able to edit any of your answers.
5. After finishing all the questions you can log out.

1. Linear Data Structures

1. $\boxed{\text{10 points}}$ **TEXT** This exercise is about stacks. Write down nine operations which can be executed on an empty stack in this order, one operation per line. After each operation, but in the same line, write down the contents of the stack after the operation, with the bottom element first.

   In addition, ensure the following:

   - At some point during your sequence of operations the stack should contain at least three items.
   - After the ninth and last operation the stack should contain at most two items.
   - Use random and different numbers from 0 to 100 as items.

2. 10 points **TEXT** Consider the following definition of the type Queue:

```
typedef struct Queue {
  int *array;
  int back;
  int front;
  int size;
} Queue;
```

If a Queue is not empty, then the `front` is the array position of the oldest item. Consider the following implementation of the operation `dequeue`:

```
int dequeue(Queue *qp) {
  if (isEmptyQueue(*qp)) {
    queueEmptyError();
  }
  qp->front = (qp->front + 1) % qp->size;
  return qp->array[qp->front - 1];
}
```

What is wrong with this implementation? Explain your answer and make suggestions how the function should be changed.

3. 10 points **TEXT** Consider the following grammar.

```
<id>       ::= <letter> { <posdigit> } [ '*' ]
<label>    ::= <letter> { <num> }
<num>      ::= <posdigit> { <digit> }
<digit>    ::= '0' | <posdigit>
<posdigit> ::= '1' | '2' | '3' | '4'
<letter>   ::= 'a' | 'b' | 'c'
```

(a) Write down two expressions which can be produced by `<id>` but not by `<label>`.
(b) Write down two expressions which can be produced by `<label>` but not by `<id>`.
(c) Write down one expression which can be produced both by `<id>` and by `<label>`.

2. Trees

1. 5 points **PDF** Draw a heap which contains all digits of your student number as elements. For example, if your student number would be 3214123 then the heap should contain seven nodes with the elements 3, 2, 1, 4, 1, 2 and 3.
Next to the drawing, also write down the array representation of the heap.

2. 10 points **PDF** Draw two search trees which contain the elements 9, 42, 50, 51, 57, 64 and your age in years. One of the trees should have at most two leaves and the other one should have at least three leaves.

3. 10 points **CODE** This problem is about binary trees. Their definition in C is as follows.

```
typedef struct TreeNode *Tree;

struct TreeNode {
  int item;
  Tree leftChild, rightChild;
};
```

Define an efficient C function with prototype `Tree singleBranch(Tree tr)` that at each node with two children removes the right child and its descendants. That is, the resulting tree should consist of a single branch.

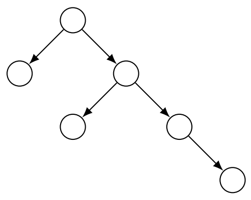
Make sure that no memory leaks occur.

You may call `freeTree(Tree t)` from the solution of Exercise 2.3 in the lecture notes.

4. $\boxed{\text{10 points}}$ **CODE** This question is about binary trees as described in the previous question.

   Define an efficient C function with prototype `int childProduct(Tree tr)` which returns the product of the number of left children and the number of right children. You may define additional helper functions to be called from `childProduct`.

   For example, given the tree below the function `childProduct` should return 6.



5. $\boxed{\text{10 points}}$ **PDF** This question is about tries and has three parts; each part relies on its successor.

   You are given the following dictionary: `{am, are, be, been, could, had, has, have}`.

   First, add your first name and your last name to the dictionary, using lower case letters. (If you have multiple first or last names, choose one of each. Also, leave out any prefixes such as "van" or "de"). Use the resulting dictionary for this whole exercise.

   Now, make the following three drawings. For each step, briefly explain any additional assumptions or decisions you made. If you decide to use a stop character to mark the end of words, please use the `$` symbol.

   (a) Draw a standard trie T for the dictionary including the words above and your first and last name.

   (b) Draw a compressed trie T' based on the standard trie T, from part a.

   (c) Draw a compact trie T" based on the compressed trie T', from part b.

   Please upload your answers for (a), (b) and (c) together as one single PDF file.

3. Graphs

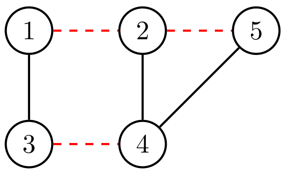
1. $\boxed{\text{5 points}}$ **PDF** Draw a connected undirected graph with 5 nodes and 8 edges which has an Euler cycle. Number the nodes and write down an Euler cycle of this graph.

2. $\boxed{\text{5 points}}$ **TEXT** Suppose we apply BFS and DFS to the same simple connected graph. What do we know about the height of the two resulting spanning trees? Explain and justify your answer!

3. $\boxed{\text{15 points}}$ **PDF** This exercise is about simple connected graphs (undirected and without weights). Suppose for each graph $G = (V, E)$ we also have a function $f\colon E \to \{\text{red}, \text{black}\}$ which assigns to each edge in G one of two labels.

   (a) Define an algorithm $\mathsf{AlternatingDistance}(G, f, v, w)$ in pseudocode that returns the length of a shortest path from v to w in G such that the edges along the path have alternating labels. If there is no path with alternating labels from v to w, the algorithm should return NoPath.

   For example, given the graph below and $v = 1$ and $w = 5$ the algorithm should return 3 and not 2 because (1,3,4,5) is a shortest path with alternating labels, whereas (1,2,5) does not have alternating labels.



   (b) What is the time complexity of your algorithm in terms of the number of nodes n and the number of edges e? Explain your answer.