Written examination in Cloud Computing

July 17th 2018

Last name: _____

First name: _____

Student number: _____

I confirm with my signature that I will process the written examination alone and that I feel healthy and capable to participate this examination.
I am aware, that from the moment, when I receive the written examination, I am a participant of this examination and I will be graded.

Signature: _____

- Provide on all sheets (including the cover sheet) your *last name*, *first name* and *student number*.
- Use the provided sheets. Own paper must *not* be used.
- Place your *ID card* and your *student ID card* on your table.
- You are allowed to use a *self prepared*, *single sided DIN-A4 sheet* in the exam. Only *handwritten originals* are allowed, but no copies.
- You are allowed to use a non-programmable calculator.
- Answers written with pencil or red pen are *not* accepted.
- Time limit: *90 minutes*
- Turn off your mobile phones!

Result:

Question:	1	2	3	4	5	6	7	8	**Σ**	Grade
Maximum points:	12	10	10	10	21	12	6	9	90	—
Achieved points:										

Last name: First name: Student number:

Question 1)

Points: .....

Maximum points: 12

Name four cloud services (only platform and infrastructure services are allowed!) you used for solving the exercise sheets. Also explain in a few words which functionality of these services you used. It should become clear why you used each single service.

Name of service	Sort of service	Explain the functionality you used and also the reason for using the service
	☐ PaaS ☐ IaaS	
	☐ PaaS ☐ IaaS	
	☐ PaaS ☐ IaaS	
	☐ PaaS ☐ IaaS	

Last name: First name: Student number:

Question 2)

Points: .....

Maximum points: 1+4+3+2=10

a) Name the functional category of OpenShift.

b) Name and explain two reasons for using OpenShift.

c) Name three software solutions / technologies that are used by OpenShift to implement its functionality.

d) Explain what a Container is and how it works.

Last name: First name: Student number:

Question 3)

Points: .....

Maximum points: 10

Explain how the Mergesort algorithm works (in a non-parallel way).

See MPI Special Challenge 2.

Last name: First name: Student number:

Question 4)

Points: .....

Maximum points: 10

Explain how the Mergesort algorithm can be implemented in a way that it sorts in parallel by using a cluster system. (*In other words: Which parts of the sorting process can be carried out in parallel by the nodes of a cluster and how is it done and what is the task of the master?*)

See the solution MPI Special Challenge 2.

Last name: First name: Student number:

Question 5 – Part 1/3) Points: .....

Maximum points: 3+3+3+3+3+3+3=21

Please fill in useful comments into the source code of this MPI Mergesort implementation. The comments should clarify what happens in the source code lines 34-36, 42, 48-49, 57-58, 71, 93-95 and 101-102.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #include <mpi.h>
5
6  void merge(int *, int *, int, int, int);
7  void mergeSort(int *, int *, int, int);
8
9  int main(int argc, char** argv) {
10     int n = atoi(argv[1]);
11     int *original_array = malloc(n * sizeof(int));
12     int numProc = atoi(argv[2]);
13     double sequentialMasterRead1, sequentialMasterRead2;
14     double sequentialTime1, sequentialTime2;
15     double parallelTime1, parallelTime2;
16
17     sequentialMasterRead1 = MPI_Wtime();
18
19     int c;
20     srand(time(NULL));
21     for(c = 0; c < n; c++) {
22         original_array[c] = rand() % n;
23     }
24
25     sequentialMasterRead2 = MPI_Wtime();
26
27     int world_rank;
28     int world_size;
29
30     // Please fill in here what the lines 34-36 are doing:
31     //
32     //
33     //
34     MPI_Init(&argc, &argv);
35     MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
36     MPI_Comm_size(MPI_COMM_WORLD, &world_size);
```

Listing 1: Mergesort with MPI (part 1/3)

Last name: First name: Student number:

Question 5 – Part 2/3) Points: .....

Maximum points: 3+3+3+3+3+3+3=21

```
37
38      // Please fill in here what the line 42 is doing:
39      //
40      //
41      //
42      int size = n/world_size;
43
44      // Please fill in here what the lines 48-49 are doing:
45      //
46      //
47      //
48      int *sub_array = malloc(size * sizeof(int));
49      MPI_Scatter(original_array, size, MPI_INT, sub_array, size, MPI_INT, 0,
        MPI_COMM_WORLD);
50
51      parallelTime1 = MPI_Wtime();
52
53      // Please fill in here what the lines 57-58 are doing:
54      //
55      //
56      //
57      int *tmp_array = malloc(size * sizeof(int));
58      mergeSort(sub_array, tmp_array, 0, (size - 1));
59
60      int *sorted = NULL;
61      if(world_rank == 0) {
62          sorted = malloc(n * sizeof(int));
63      }
64
65      parallelTime2 = MPI_Wtime();
66
67      // Please fill in here what the line 71 is doing:
68      //
69      //
70      //
71      MPI_Gather(sub_array, size, MPI_INT, sorted, size, MPI_INT, 0,
        MPI_COMM_WORLD);
72
73      sequentialTime1 = MPI_Wtime();
```

Listing 2: Mergesort with MPI (part 2/3)

Last name: First name: Student number:

Question 5 – Part 3/3)

Points: .....

Maximum points: 3+3+3+3+3+3+3=21

```
74
75      if(world_rank == 0) {
76          int *other_array = malloc(n * sizeof(int));
77          mergeSort(sorted, other_array, 0, (n - 1));
78
79          free(sorted);
80          free(other_array);
81      }
82
83      sequentialTime2 = MPI_Wtime();
84
85      free(original_array);
86      free(sub_array);
87      free(tmp_array);
88
89      // Please fill in here what the lines 93-95 are doing:
90      //
91      //
92      //
93      if(world_rank == 0) {
94          printf("%i \t %.3f \t\t %f \t %f \t\t %f \n",numProc, (
        sequentialTime2 - sequentialMasterRead1), (sequentialMasterRead2 -
        sequentialMasterRead1), (parallelTime2 - parallelTime1), (sequentialTime2
         - sequentialTime1) );
95      }
96
97      // Please fill in here what the lines 101-102 are doing:
98      //
99      //
100     //
101     MPI_Barrier(MPI_COMM_WORLD);
102     MPI_Finalize();
103 }
104
105 /********** Merge Function **********/
106 void merge(int *a, int *b, int l, int m, int r) { ... }
107
108 /********** Recursive Merge Function **********/
109 void mergeSort(int *a, int *b, int l, int r) { ... }
```

Listing 3: Mergesort with MPI (part 3/3)

Last name: First name: Student number:

Question 6)

Points: .....

Maximum points: 12

This two diagrams show the total execution time of the Mergesort application from question 5 for two different problem sizes = number of integer values to be sorted.

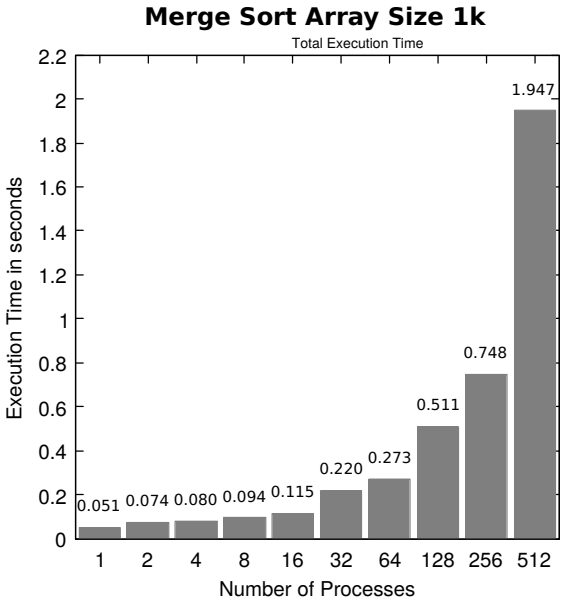

Figure 1: Problem Size = 1,000 values

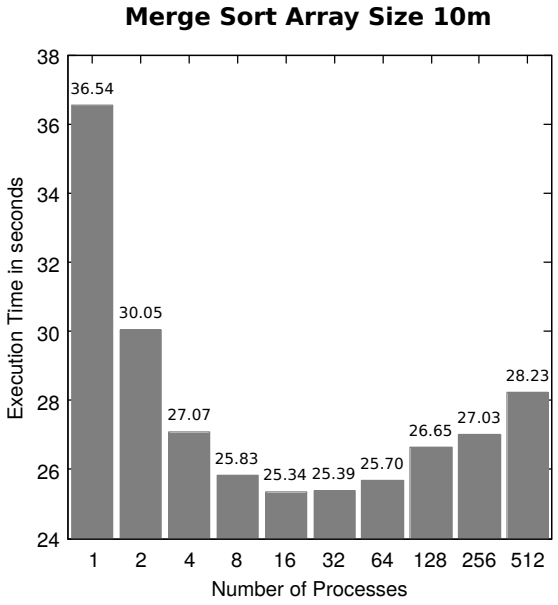

Figure 2: Problem Size = 10,000,000 values

The two diagrams demonstrate two fundamental laws and limitations of parallel computing.

a) Name the two fundamental laws and limitations of parallel computing which are relevant here.

b) Explain the two fundamental laws and limitations of parallel computing by using the two diagrams.

Last name: First name: Student number:

Question 7)

Points: .....

Maximum points: 1+1+1+1+1+1=6

a) Explain what an Active/Active-Cluster is.

b) Explain what an Active/Passive-Cluster is.

c) Explain what the meaning of Failover is.

d) Explain what the meaning of Failback is.

e) Explain what a Beowulf Cluster is.

f) Explain what a Wulfpack Cluster is.

Last name: First name: Student number:

Question 8)

Points: .....

Maximum points: 9

a) During the guest lecture from Novatec on June 13th, the six quality goals from the ISO/IEC 9126 standard, which classifies software quality, have been discussed. Name three of them. *Just name them! No explanation is required.*

b) During the guest lecture from Novatec on June 13th, the twelve factors from the twelve-factor app, which are recommended for building software-as-a-service apps, have been discussed. Name and explain (in short!) six of them.