Algorithms and Data Structures
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: CH-231-A
May 11th, 2020

Practice Sheet

First Name: ....

Last Name: ....

Matriculation Number: ....

- Read all the following points before proceeding to the solution.
- Write immediately your name on this sheet.
- Write clearly.
- Books, slides, notes, other documents or electronic devices are not allowed.
- If you need more space to solve the exercises you may use also the back of each page.
- Read carefully the questions and strictly adhere to the requirements. Write down all intermediate steps and computations.
- You have 120 minutes to solve the problems of this final exam.
- Any attempt to cheat leads to an immediate fail.
- By signing this sheet you imply you read and understood all of the above.

Signature: ....

%	0.00 - 39.49	39.50 - 44.49	44.50 - 49.49	49.50 - 54.49
Grade	5.0	4.7	4.3	4.0

%	54.50 - 59.49	59.50 - 64.49	64.50 - 69.49	69.50 - 74.49
Grade	3.7	3.3	3.0	2.7

74.50 - 79.49	79.50 - 84.49	84.50 - 89.49	89.50 - 94.49	94.50 - 100.00
2.3	2.0	1.7	1.3	1.0

Problem P.1 *STL Containers* (1 point)

Select the corrrect answer for the following question:
Which list is the longest list with containers in which you can you use the `insert` method?

(1) `vector, set, map`

(2) `vector, list, deque`

(3) `set, multiset, map, multimap`

(4) `vector, list, deque, set, multiset, map, multimap`

(5) None of the above

Problem P.2 *Templates I* (2 points)

Write down the output of the following C++ program:

```
#include <iostream>
using namespace std;

template <class T>
class Test {
    private:
        T val;
    public:
        static int count;
        Test()  {   count++;   }
    };

template<class T>
int Test<T>::count = 0;

int main() {
        Test<int> a;
        Test<int> b;
        Test<double> c;
        cout << Test<int>::count   << endl;
        cout << Test<double>::count << endl;
        return 0;
}
```

Problem P.3 *Templates II* (2 points)

Create the C++ template function named `multiples` so that it has three parameters: `sum`, `x`, and `n`. The function should use the `sum` variable to "return by reference" $sum = 1{+}x{+}2x{+}3x{+}...{+}nx$.

Problem P.4 *Search in Data Structures* (2 points)

Select one or multiple correct answers for the following statement:
The following data structures can perform a search operation in time of O($n \log n$):

(1) Heap

(2) Max-heap

(3) Queue

(4) Binary search tree

(5) Red black tree

Problem P.5 *Asymptotic Time Complexity* (2 points)

Consider the recurrence $T(n) = 3T(n/2) + \Theta(n)$ with $T(1) = \Theta(1)$.
Select True or False for each of the following statements:

True/False $T(n) \in \Theta(n \log n)$

True/False $T(n) \in \Theta(n^3 \log n)$

True/False $T(n) \in \Theta(\log n)$

True/False $T(n) \in \Theta(n^3)$

True/False $T(n) \in \Theta(3n \log n)$

Problem P.6 *Fast Multiplication* (4 points)

Recall the divide and conquer formula of fast exponentiation of a number a $\in \mathbb{R}$ to a power $n \in \mathbb{N}$:

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & \text{n is even} \\ a^{\lfloor n/2 \rfloor} \cdot a^{\lfloor n/2 \rfloor} \cdot a & \text{n is odd} \end{cases}$$

a) Give a similar formula for a divide and conquer approach to the fast multiplication problem of a $\in \mathbb{R}$ and n $\in \mathbb{N}$.

b) Write a pseudocode implementation of a function that uses the derived formula to multiply a and n.

c) Derive and prove the asymptotic time complexity (upper and lower bound) of your algorithm.

Problem P.7 *Binary Search Tree* (4 points)

Study the given code. The goal of the transform function is to convert a binary search tree into a new one that meets the following requirements:

- It is still a binary search tree.
- Every vertex has only one child.
- If a vertex is the left child of it's parent it can't have a left child. If it is a right child it can't have a right child.
- It contains the exact same nodes as the initial tree.

Implement or write pseudocode for the `transform` function using the other given functions. The root of the resulting tree should be returned through the given parameter of the function

```
struct node{
    int data;

    node *left;
    node *right;
    node *parent;
};

void replace(node **root, node *x, node *y){
    if(x == *root)
        *root = y;
    else {
        if(y != NULL)
            y->parent = x->parent;

        if(x->parent->left == x)
            x->parent->left = y;
        else
            x->parent->right = y;
    }
}
```

```
node* find_min(node **root){
    node *min   = NULL;
    node *tmp   = *root;

    while(tmp != NULL)
        min = tmp,
        tmp = tmp->left;

    return min;
}

node* extract_min(node **root){
    node *min = find_min(root);

    if(min == NULL)
        return NULL;

    replace(root, min, min->right);
    min->right = NULL;

    return min;
}

node* find_max(node **root){
    node *max   = NULL;
    node *tmp   = *root;

    while(tmp != NULL)
        max = tmp,
        tmp = tmp->right;

    return max;
}

node* extract_max(node **root){
    node *max = find_max(root);

    if(max == NULL)
        return NULL;

    replace(root, max, max->left);
    max->left = NULL;

    return max;
}

void transform(node **root);
```

Problem P.8 *Quicksort* (3 points)

Write down the complete pseudocode for randomized Quicksort including functions which are required for partitioning.

Problem P.9 *Red Black Trees* (2 points)

Make an example of a red black tree with the height of 3. You do not need to draw a tree. Just list the nodes per level from left to right and indicate the parent, left or right child, the value and the color for each node.

Problem P.10 *Greedy Algorithm* (5 points)

Consider the following minimizing lateness problem for multiple jobs/tasks:

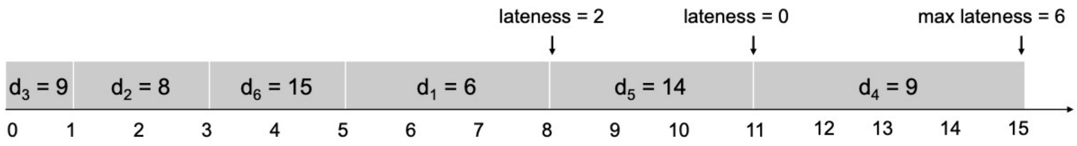
- Only one job can be processed at a time.
- Job j requires t_j units of processing time and is due at time d_j.

- If j starts at time s_j, it finishes at time $f_j = s_j + t_j$.
- The lateness: $l_j = max\{0, f_j - d_j\}$
- Develop a greedy algorithm to schedule all jobs to minimize the *maximum* lateness $L = max\ l_j$.

Here is an example:

Ex:

	1	2	3	4	5	6
t_j	3	2	1	4	3	2
d_j	6	8	9	9	14	15



a) Prove that greedy algorithms that make the greedy choices of *shortest processing time first* t_j and *smallest slack* $d_j - t_j$ both fail to produce a globally optimal solution.

b) Develop the greedy algorithm in pseudocode that makes the correct greedy choice. Use the example in the problem to verify your solution (the optimal solution has a maximum lateness of 1).

Problem P.11 *Graph Algorithms* (3 points)

Run the Breadth-First-Search algorithm starting from node 0 on the following example graph. Write down the detailed steps below.