Roland Benz, rolben, 97923163

Algorithmic Complexity and Correctness [25 points]
Task 1. [20 points]

b) Do an exact analysis of the running time of the algorithm.

Algo:
Input: array A[1..n] of length n, 1 ≤ k ≤ n
Output: print the maximum value of every contiguous subarray of size k

findKMax(A, k, n)	Duration	Number of executions
for i = 1 to n − k + 1 do	c1	(n-k+1)
max = A[i];	c2	(n-k+1)
for j = 1 to k − 1 do	c3	(n-k+1)*(k-1)
if A[i + j] > max then	c4	(n-k+1)*(k-1)
max = A[i + j];	c5	(n-k+1)*0..(k-1)
print(max)	c6	(n-k+1)

Runtime:
T(n)=(c1+c2+c6)*(n-k+1)+(c3+c4)*(n-k+1)*(k-1)+c5*(n-k+1)*0..(k-1)

c) Determine the best and the worst case of the algorithm. What is the running time and asymptotic complexity in each case?
(n-k+1)*(k-1)= n*k+2*k-n-k^2-1
Best case:
Running time: T(n)=(c1+c2+c6)*(n-k+1)+(c3+c4)*(n-k+1)*(k-1)
Asymptotic complexity: T(n)=O(n*k - k^2)
Worst case:
Running time: T(n)=(c1+c2+c6)*(n-k+1)+(c3+c4+c5)*(n-k+1)*(k-1)
Asymptotic complexity: T(n)=O(n*k - k^2)

d) What influence has the parameter k on the asymptotic complexity?
If k and n of same order: T(n)=O(1)
If k<<n: T(n)=O(n)

Task 2. [5 points]

Given an unsorted array A[1..n] that contains only numbers 0, 1, and 2, the following algorithm rearranges the elements of A, such that all occurrences of 0 come before all occurrences of 1 and all occurrences of 1 come before all occurrences of 2.

State a loop invariant for the algorithm partitionValues and show that it is correct.

Algo: PARTITIONVALUES(A, n)

Input: array $A[1..n]$ of length n
Output: array $A[1..n]$ rearranged

```
k = 1;
l = 1;
m = n;
while l ≤ m do
    if A[l] = 0 then
        swap(A[k], A[l]);
        k = k + 1;
        l = l + 1;
    else if A[l] = 1 then
        l = l + 1;
    else
        swap(A[l], A[m]);
        m = m − 1;
```

l is the iterator
k is the left border between 0 and 1
m is the right border between 1 and 2
Invariants:
(0) progression is guaranteed:
->either l moves to the right or m to the left
(1) for all indices i < k: A[i]=0
(2) for all indices i >m: A[i]=2

Start:
(1) ok because i is out of bound
(2) ok because i is out of bound
Maintenance:
(1) ok because if statement is moving 0s to the left of k_new
(2) ok because else statement is moving 2s to the right of m_new
Termination:
(1) ok because at l=m last step is correct then stop
(2) ok because at l=m last step is correct then stop

Asymptotic Complexity [3 points]
Task 3. [3 points]

Calculate the asymptotic tight bound for the following functions and rank them by their order of growth (lowest first). Clearly work out the calculation steps in your solution.

Roland Benz, rolben, 97923163

f1(n) = log(pi*n) + log(100^log(n)) log(pi*n) + log(n)*log(100) Theta(log(n))	$f_1(n) = \log(\pi n) + \log(100^{\log n})$ $f_2(n) = 10^{\lg 20} n^4 + 8^{229} n^3 + 20^{231} n^2 + 128 n \log n$ $f_3(n) = \log n^{2n+1}$ $f_4(n) = 101^{\sqrt{n}}$ $f_5(n) = 2^n + \sqrt{n}$ $f_6(n) = (n+1)!$
f2(n) = 10^(lg(20)) * n^4 + 8^229 * n^3 + 20^231 * n^2 + 128*n * log(n) Theta(n^4)	
f3(n) = log(n^(2n+1)) (2n+1)* log(n) Theta(n*log(n))	
f4(n) = 101^(n^0.5) Theta(101^(n^0.5))	
f5(n) = 2n + (n^0.5) Theta(2^n)	
f6(n) = (n + 1)! Theta((n+1)!)	

Ranking:
f1(n)<f3(n)<f2(n)<f4(n)<f5(n)<f6(n)

Special Case Analysis [15 points]
Task 4. [15 points]

Given two strings A and B, develop an algorithm that checks if B is a permutation of A. For example, if B = "aabb" and A = "baba", the return value will be TRUE. If B = "ab" and A = "baba", the return value will be FALSE.

a) Specify all the special cases that need to be considered and provide examples of the input data for each of them.

1.Size(A) = Size (B)
If e.g. A=[...] and B=[……..] then stop

2.Size(A) = Size (B) = 0
If A=[] and B=[] return true

3.Size(A) = Size (B) = 1
If A=[.] and B=[.] compare values and return

4.CharacerIn(A) = CharacterIn(B)
If A=[xaabbccaabbccaabbcc] and B=[aaabbccaabbccaabbcc] x not found in B then stop

5.CountOfCharacterIn(A) = CountOfCharacterIn(B)
If A=[aabbccaabbccaabbcc] and B=[aaabbccaabbccaabbcc] count occurrences and return

Recurrences [12 points]
Task 5. [6 points]

Recurrence
T (n)=1 if n = 1
T (n) =T (n/6) + T (n/2) + n if n > 1

a) Draw a recursion tree and use it to estimate the asymptotic upper bound of T (n). Include the tree-based calculations that led to your estimate. [3 points]

				n				n
			n/6	n/2				n4/6
		n/36	n/12	n/12	n/4			n16/36
n/216	n/72	n/72	n/24	n/72	n/24	n/24	n/8	n64/216

T(n/6) = T (n/36) + T (n/12) + n/6
T(n/2)= T (n/12) + T (n/4) + n/2
T (n/36) =T (n/216) + T (n/72) + n/36
T (n/12) =T (n/72) + T (n/24) + n/12
T (n/4) =T (n/24) + T (n/8) + n/4

T(n)=n+n*sum((4/6)^i, i=1..inf)
=n+n*(1/(1-4/6))
=4n
=O(n)

b) Prove the correctness of your estimate using the substitution method. [3 points]
Recurrence: T (n) =T (n/6) + T (n/2) + n
Guess: T(n)=O(n)
Show that: T(n)<=cn
T (n) =T (n/6) + T (n/2) + n<=cn/6+cn/2+n=4cn/6+n<=cn für c>=3

Roland Benz, rolben, 97923163

Task 6. [6 points] Calculate the asymptotic tight bound of the following recurrences. If the Master Theorem can be used, write down a, b, f (n) and the case (1-3).

<table>
<tr><td>1. a=3, b=9, f(n)=32*n^0.5=O(n^c) mit c=0.5
log9(3)=0.5
Case 2: T(n)=Theta(n^0.5 *log10(n))</td><td rowspan="4">1. $T(n) = 3T(\frac{n}{9}) + 32\sqrt{n}$
2. $T(n) = 16T(\frac{n}{4}) + n^3$
3. $T(n) = \sqrt{2}T(\frac{n}{2}) + \log n$
4. $T(n) = T(n-2) + n$</td></tr>
<tr><td>2. a=16, b=4, f(n)=n^3=O(n^c) mit c=3
log4(16)=2
Case 3: T(n)=Theta(n^3)</td></tr>
<tr><td>3. a=2^0.5, b=2, f(n)=log(n)=O(n^c) mit c=log10 (log10 (n))/log10 (n)
für n=1 Mio, c=0.129
log2(2^0.5)=0.5
Case 1: T(n)=Theta(n^0.5)</td></tr>
<tr><td>4. T(n)=T(n-6)+n-4+n-2+n
=T(n-2*i)+n*i-sum(-2*j, j=1,i-1), imax=(n-1)/2
=T(1)+n*(n-1)/2+?
=Theta(n^2)</td></tr>
</table>

Master Method:

Master Method is a direct way to get the solution. The master method works only for following type of recurrences or for recurrences that can be transformed to following type.

`T(n) = aT(n/b) + f(n) where a >= 1 and b > 1`

There are following three cases:

1. If f(n) = $\Theta(n^c)$ where c < $Log_b a$ then T(n) = $\Theta(n^{Log_b a})$

2. If f(n) = $\Theta(n^c)$ where c = $Log_b a$ then T(n) = $\Theta(n^c Log\ n)$

3.If f(n) = $\Theta(n^c)$ where c > $Log_b a$ then T(n) = $\Theta(f(n))$