**LAB1**

**Jimmy Palma**

**610756**

**Problem 1.** Which of the following functions are increasing? eventually nondecreasing?

1. **f(x) = -xˆ2**

f ' (x) = -2x

Increase: ]−∞, -1]

Decrease: [1, ∞[

1. **f(x) = xˆ2 + 2x + 1**

f ' (x) = 2x + 2

Increase: [-1, -∞[

Decrease: ]-∞, -1 ]

1. **f(x) = xˆ3 + x**

f ' (x) = 3xˆ2 + 1

Increase: ]-∞,∞[

**Problem 2.** Use the limit definitions of complexity classes given in class to decide whether each of the following is true or false, and in each case, prove your answer.

1. **4n³ + n is θ(nˆ3)**

Lim (4n³+n)/n³

n→∞

Lim (4+1/n²)/1

n→∞

Lim (4+ 0)/1

n→∞

= 4

1. **Log n is o(n)**

Lim log(n) / n

n→∞

= 0

1. **2ˆn is ω(n²)**

Lim **n² / 2 ^ n**

n→∞

Lim 2**n / 2 ^ n ln 2**

n→∞

= 0

1. **2 ˆn is o(3 ^n)**

Lim 2 ^ n/ 3 ^ n

n→∞

Lim (2 / 3) ^ n

n→∞

= 0

**Problem 3.** Show that for all n > 4, 2^ n < n!. Hint: Use induction.

Basic step: n = 5, 2 ^ 5 < 5!

32 < 120

Induction step: 2^(n + 1) < (n + 1)!

2(2^n) < n!(n+1)

2 < n + 1 like n > 4

2 < 6

TRUE

**Problem 4.** GCD Problem: Given two positive integers m, n, is there a positive integer d that is a factor of both m and n and that is bigger than or equal to every integer d’ that is also a factor of m and n?

Write a Java method int gcd(int m, int n) which accepts positive integer inputs m; n and outputs the greatest common divisor of m and n.

Examples

f m = 12 and n = 42, return 6

If m = 7 and n = 9, return 1

///////// code ////////

public static int gcd(int m, int n) {

int lowest = m < n ? m : n;

for(int i = lowest; i > 0; i--) {

if(m % i == 0 && n % i == 0) {

return i;

}

}

return 0;

}

**Problem 5.** Implement the following Java method.  
 public static int secondSmallest(int[] arr) {

if(arr==null || arr.length < 2) {  
 throw new IllegalArgumentException("Input array too small");

}

//implement

}

This method returns the second smallest element of the input array.

Examples

If input is [1, 4, 2, 3], return 2.

If input is [3, 3, 4, 7], return 3. (Smallest is 3, and second smallest is 3.)

If input is [9], your program will throw an exception.

///////////// code /////////

public static int secondSmallest(int[] arr) {

if(arr==null || arr.length < 2) {

throw new IllegalArgumentException("Input array too small");

}

int smallest = arr[0];

int secondSmallest = arr[1];

if(smallest > secondSmallest) {

int temp = smallest;

smallest = secondSmallest;

secondSmallest = temp;

}

for(int i = 2; i < arr.length; ++i) {

int next = arr[i];

if(next < smallest) {

secondSmallest = smallest;

smallest = next;

}

else if(next < secondSmallest) {

secondSmallest = next;

}

}

return secondSmallest;

}

**Problem 6.** SubsetSum Problem: given a set S = {s0, s1,s2, ..., sn-1}of positive integers and a non-negative integer k, is there a subset T of S so that the sum of the integers in T equals k?

Formulate your own procedure for solving the SubsetSum. Think of it as a Java method subsetsum that accepts input S and k, and outputs a subset T of S with the property that the sum of the elements in T is k if such a T exists, or null if no such T can be found.

Examples

If S is [1, 3, 9, 4, 8, 5] and k = 21, return [9,4,8] (since 9 + 4 + 8 = 21)

If S is [1, 3, 9] and k = 5, return null (since no such subset can be found)

If S is [1, 3, 9, 4, 8, 5] and k = 0, return [] (since the sum of the empty set is 0)

///////// code ////////

public static List<Integer> subsetSum(List<Integer> ints, int testInt) {

List<List<Integer>> finalList = *getPowerSet*(ints);

for (List<Integer> list : finalList) {

if(testInt == list.stream().reduce(0, (x, y)-> x + y)){

return list;

}

}

return null;

}

private static List<List<Integer>> getPowerSet(List<Integer> ints) {

int resultSize = (int) Math.*pow*(2, ints.size());

List<List<Integer>> p = new ArrayList<List<Integer>>(resultSize);

p.add(new ArrayList<Integer>(0));

for (Integer itemFromOriginalSet : ints) {

int startingResultSize = p.size();

for (int i=0; i<startingResultSize; i++) {

List<Integer> px = p.get(i);

List<Integer> newSet = new ArrayList<Integer>(px);

newSet.add(itemFromOriginalSet);

p.add(newSet);

}

}

return p;

}