UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Analysis and Design of Algorithms

CS2102
Divide and Conquer Practice
2020-II

Submission deadline: 15 Oct, 20:05
Number of questions: 5

- Write your C++ code inside the *answers* folder in order to generate a single PDF file. The *problem*{*1,2,3,4,5*}*.cpp* files should include the functions required by the problem and the main function. No need to create header or other source files per problem.
- Read the questions carefully and write your answers clearly. Answers that are not legible will not have any score.
- Notes are allowed. To compile this file you can use the command latexmk -pdf -shell-escape main.tex
- Consider edge cases properly, any file that doesn't compile or doesn't met the requirements will not have any grade (clang++ or g++ are preferred).
- **STD compiler flag**: -std=c++2a (replace this by your actual configuration)

Outcomes:

a. Apply appropriate mathematical and related knowledge to computer science.

b. Analyze problems and identify the appropriate computational requirements for its solution.

Problem 1 (Outcomes a) - 5 points

Write a divide-and-conquer $\Theta(n * log(n))$ program to measure how far a sequence of n numbers is from being sorted in descending order.

Intuition: If we measure the number of changes we need to do in a list of numbers to be sorted in ascending order we need to count the number of pairs i, j such that $A[i] > A[j]$ and $i < j$. That count will give us the measure of how far a sequence of numbers is from being sorted. For example, using the previous definition the similarity measure for the array $\{2,4,1,3,5\}$ is 3 since (2,1), (4,1) and (4,3) are out of order.

```
#include <iostream>
#include <vector>

int problem1(std::vector<int> && numbers) {
    return 0;
}
```

```
int main() {
    std::cout << "Problem 1 - Similarity\n";

    std::vector<int> numbers = {2,4,1,3,5}; // This is just an input
     ↪  example, consider all the ones you think are relevant ( consider
     ↪  the same for the remaining problems )
    std::cout << problem1(std::move(numbers)) << "\n";
}
```

Problem 2 (Outcomes a) - 3 points

Write a divide-and-conquer $\Theta(n * log(n))$ program that counts the total number of times that a number k appears in a sequence of numbers.

```
#include <iostream>
#include <vector>

int problem2(std::vector<int> &&numbers, int k){
    return 5;
}

int main(){
    std::cout << "Problem 2 - Occurrences\n";

    std::vector<int> numbers = {5,5,5,5,5};
    int k = 5;
    std::cout << problem2(std::move(numbers), k) << std::endl;
}
```

Problem 3 (Outcomes a) - 5 points

Consider the Volatile Chemical Corporation investing problem described in Cormen's Chap 04. Write a divide-and-conquer $\Theta(n * log(n))$ program that receives an input array of n numbers representing the stock prices of each day and returns the days that maximizes our profit when buying and selling stocks.

```
#include <iostream>
#include <vector>

std::pair<int,int> problem3(std::vector<int> && stockPrices){

    return std::make_pair(0,1);
}
```

```
int main() {
    std::cout << "Problem 3 - Volatile investment\n";

    std::vector<int> stocks =
     ↪  {100,113,110,85,105,102,86,63,81,101,94,106,101,79,94,90,97};
    auto [buy, sell] = problem3(std::move(stocks));

    printf("Buying at day %d\nSelling at day %d\n", buy, sell);
}
```

Problem 4 (Outcomes a) - 3 points

Write a lineal (1pt) and a divide-and-conquer (2pt) $\Theta(log(n))$ program that compute x^n considering x as the base and n as the exponent.

```
#include <iostream>

long problem4(long base, long power){
    return 0;
}

int main() {
    std::cout << "Problem 4 - Powering\n";

    long base = 2;
    long power = 2;
    printf("%ld power of %ld is %ld\n", base, power, problem4(base,
     ↪  power));
}
```

Problem 5 (Outcomes a, b) - 4 points

Write a recursive insertion sort algorithm (2pt) that receives an unsorted array of numbers, then express that algorithm using a recurrence relation and approximate T(n) (2pt).

```
#include <iostream>

int main() {
    std::cout << "Problem 5 - Recursive Insertion Sort\n";
}
```

$T(n) = 2T(n/2) + n$ (replace these lines with your answer)
$T(n) = O(n)$