Algorithms & Data Structures
Exercise Sheets

Weeks 4-5: Matrices and array searching

Submit your solutions via the peergrade page on Brightspace, Week 4-5. Please make sure to submit your solutions **before the deadline** (typically a Friday morning, but check the deadline in peergrade). If an exercise requires you to write C++ code, then create a new C++ file for each exercise (i.e. do not combine exercises into a single file). You can hand-in in pairs of two.

Note that you also needs to review both other students' and your own solution. If you handed in as a group, you will have to review your submission individually (I kindly ask you to do it individually, that gives you the best learning outcome). The deadline for revievs is typically Thursday after the hand-in at mid-night (BUT check the actual deadline in peergrade).

Exercises

(1) Rewrite the following code so that it uses stl structure(s), iterator(s) and algorithm(s)

```c
#include <stdio.h>

// Naive method to find a pair in an array with a given sum
void findPair(int arr[], int n, int target) {
  // consider each element except the last
  for (int i = 0; i < n - 1; i++) {
    // start from the i'th element until the last element
    for (int j = i + 1; j < n; j++) {
      // if the desired sum is found, print it
      if (arr[i] + arr[j] == target) {
        printf("Pair found (%d, %d)", arr[i], arr[j]);
        return;
      }
    }
  }
  // we reach here if the pair is not found
  printf("Pair not found");
}

int main(void)
{
  int arr[] = { 8, 7, 2, 5, 3, 1 };
  int target = 10;

  int n = sizeof(arr)/sizeof(arr[0]);
  findPair(arr, n, target);
  return 0;
}
```

(2) Given a monotonically increasing function $f(x)$ on the interval $[a, b]$ and a real number C, write a program that finds x such that $f(x) = C$ with logarithmic worst-case complexity. Test your code with at least two choices of f. **Hint:** A monotonically increasing function is such that $f(x) \leq f(y)$ if $x \leq y$, $\forall x, y \in [a, b]$.

For example, you should implement a function with the following signature (`f` is a pointer to a monotonically increasing function):

```
double search(double a, double b, double c, double (*f)(double x));
```

(3) Implement a **recursive** algorithm for finding the maximum and minimum elements in an array A of N elements. State the complexity in terms of a recurrence relation and solve it for the worst-case analysis.

(4) Add a `transpose()` member function to the `Matrix` class that replaces a matrix with its transpose. Try to do this without the use of a temporary matrix. What is the resulting worst-case time complexity in Θ notation?

(5) Write a function that takes a *sorted* N by N matrix already stored in memory and decides if a number x is in the matrix with worst-case complexity $O(N)$ (linear in **one** of the dimensions). Assume that each individual row is increasing from left to right and each individual column is increasing from top to bottom. Justify the achieved complexity.