Last name(s)	**Name**	**ID**

Midterm EDA exam **Length: 2.5 hours** **05/11/2018**

- *The exam has 4 sheets, 8 sides and 4 problems.*
- *Write your full name and ID on every sheet.*
- *Write your answers to all problems in the exam sheets within the reserved space.*

Problem 1 **(2 points)**

For each of the following statements, mark with an X the corresponding cell depending on whether it is true or false.

Note: Each right answer will add 0.2 points; each wrong answer will subtract 0.2 points, except in the case that there are more wrong answers than right ones, in which the grade of the exercise will be 0.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
TRUE										
FALSE										

(1) The cost in time of quicksort in the worst case is $\Theta(n^2)$.

(2) The cost in time of quicksort in the average case is $\Theta(n^2)$.

(3) Any algorithm that computes the sum $x + y$ of two naturals x, y of n bits each, must have cost in time $\Omega(n)$.

(4) There exists an algorithm that computes the sum $x + y$ of two naturals x, y of n bits each, in time $O(n)$.

(5) Any algorithm that computes the product $x \cdot y$ of two naturals x, y of n bits each, must have cost in time $\Omega(n^2)$.

(6) There exists an algorithm that computes the product $x \cdot y$ of two naturals x, y of n bits each, in time $O(n^2)$.

(7) Any algorithm that, given an integer $k \geq 0$ and a matrix A of $n \times n$ integer numbers, computes the matrix A^k must do $\Omega(k)$ products of matrices.

(8) $2^{2n} \in O(2^n)$.

(9) $2n \in O(n)$.

(10) $\log(2n) \in O(\log n)$.

This side would be intentionally blank if it were not for this note.

Last name(s) **Name** **ID**

Problem 2 **(3 points)**

Given two matrices of Booleans A and B of size $n \times n$, we define their *logical product* $P = A \cdot B$ as the $n \times n$ matrix which at the coefficient of the i-th row and j-th column ($0 \leq i, j < n$) contains:

$$p_{ij} = \bigvee_{k=0}^{n-1} (a_{ik} \wedge b_{kj})$$

(a) (0.5 pts.) Consider the following function for computing the logical product:

```
vector<vector<bool>> product1(const vector<vector<bool>>& A,
                              const vector<vector<bool>>& B) {
  int n = A.size();
  vector<vector<bool>> P(n, vector<bool>(n, false));
  for (int i = 0; i < n; ++i)
    for (int j = 0; j < n; ++j)
      for (int k = 0; k < n; ++k)
        P[i][j] = P[i][j] or (A[i][k] and B[k][j]);
  return P;
}
```

What is the cost in time in the worst case in terms of n? The cost is $\Theta($ $)$.
Give an example of worst case.

(b) (1 pt.) Consider the following alternative for computing the logical product:

```
vector<vector<bool>> product2(const vector<vector<bool>>& A,
                              const vector<vector<bool>>& B) {
  int n = A.size();
  vector<vector<bool>> P(n, vector<bool>(n, false));
  for (int i = 0; i < n; ++i)
    for (int j = 0; j < n; ++j)
      for (int k = 0; k < n and not P[i][j]; ++k)
        if (A[i][k] and B[k][j]) P[i][j] = true;
  return P;
}
```

What is the cost in time in the best case in terms of n? The cost is $\Theta(\boxed{})$.

Give an example of best case.

What is the cost in time in the worst case in terms of n? The cost is $\Theta(\boxed{})$.

Give an example of worst case.

(c) (1.5 pt.) Explain at a high level how to implement a function for computing the logical product of matrices that is more efficient asymptotically in the worst case than those proposed at sections (a) and (b). What is its cost in time in the worst case?

Last name(s) **Name** **ID**

Problem 3 **(2 points)**

Consider the following function:

```
int mystery_rec(int n, int l, int r) {
  if (r == l+1) return l;
  int m = (l+r)/2;
  if (m*m ≤ n) return mystery_rec(n, m, r);
  else         return mystery_rec(n, l, m);
}

int mystery(int n) {
  return mystery_rec(n, 0, n+1);
}
```

(a) (1 pt.) Given an integer $n \geq 0$, what does function *mystery* compute?

(b) (1 pt.) What is the cost in time of *mystery*(n) in terms of n? The cost is $\Theta($ $)$

Justify your answer.

This side would be intentionally blank if it were not for this note.

Last name(s) **Name** **ID**

Problem 4 **(3 points)**

Consider the problem of, given a vector of integers (possibly with repetitions), sorting it in increasing order. The following costs are in time and are asked in terms of n, the size of the vector.

(a) (0.25 pts.) What is the cost of the insertion sort algorithm in the best case? The cost is $\Theta($ $)$.

(b) (0.25 pts.) What is the cost of the insertion sort algorithm in the worst case? The cost is $\Theta($ $)$.

(c) (0.25 pts.) What is the cost of the mergesort algorithm in the best case? The cost is $\Theta($ $)$.

(d) (0.25 pts.) What is the cost of the mergesort algorithm in the worst case? The cost is $\Theta($ $)$.

(e) (0.75 pts.) Fill the gaps of the function *my_sort* defined below, so that given a vector of integers v, it sorts it increasingly:

```
void my_sort(vector<int>& v) {
  int n = v.size();
  double lim = n * log(n);
  int c = 0;
  for (int i = 1; i < n; ++i) {
    int x = v[i];
    int j;
    for (j = i; j > 0 and [          ]; --j) {
      v[j] = [          ];
      ++c;
    }
    v[j] = [          ];
    if (c > lim) {
      merge_sort(v);
      return;
    }
  }
}
```

The auxiliary function *merge_sort* is an implementation of the mergesort algorithm, and function *log* computes the neperian logarithm (that is, in base e).

(f) (1.25 pts.) What is the cost of *my_sort* in the best case? The cost is $\Theta(\boxed{})$.

What is the cost of *my_sort* in the worst case? The cost is $\Theta(\boxed{})$.

Justify your answers and give one example of best case and one of worst case.