Dear student,

Welcome to the Midterm Exam of «Data Structures and Algorithms» course!

This document contains 4 questions: question F, question N, question P, and question S. Please read each question carefully and provide the solutions as directed.

Solutions should be written by hand on paper, using pen or pencil. Be sure to make your writing clear and readable. After writing down the solutions, you should take a high quality photo of every page and submit those photos as a ZIP archive on Moodle.

In case Moodle is not available, you should submit the archive via email to n.kudasov@innopolis.ru using subject "DSA Final Exam: <Your Name>".

Please make sure to leave at least 5 minutes before the end of the exam to properly submit your solutions! **Submissions sent after the end of the exam will not be accepted.**

1 Question N (25 points)

1.1 Complexity analysis

Consider the following declarations given in a C-like programming language. Give a tight asymptotic time complexity for the worst-case running time of function `f` in terms of the size of input array `A`. Justify your answer.

```
1  function f(A) { return a[h(A, 1, 0)]; }
2
3  function h(A, m, k) {
4      if (m < A.length) {
5          x := h(A, 3*m, k);
6          y := h(A, 3*m, m + k);
7          z := h(A, 3*m, m + 2*k);
8          r = k;
9          for (i = k + m; i < A.length; i = i + m) {
10             if (A[r] > A[i]) { r = i; }
11         }
12         return x + y + z + r;
13     } else { return min(k+1, A.length); }
14 }
```

1.2 Algorithmic Strategies and Master Theorem

Answer the following questions about algorithmic strategies:

1. True or False? A brute force algorithm searching for a value in an unsorted sequence has asymptotic time complexity of $\Omega(n \cdot 2^n)$.
2. True or False? Any recurrence relation of the form $T(n) = aT(n/b) + f(n)$ where $a \geq 1, b > 1$ and $f(n)$ is an asymptotically non-negative function, can be solved using at most one of the three cases of the master theorem.
3. Consider recurrence relation $T(n) = 3T(n/27) + \sqrt[3]{n+1}$. Applying the master theorem we get:
   (a) $T(n) = \Theta(\sqrt{n})$;
   (b) $T(n) = \Theta(\sqrt{n} \cdot \log n)$;
   (c) $T(n) = \Theta(\sqrt[3]{n})$;
   (d) $T(n) = \Theta(\sqrt[3]{n} \cdot \log n)$;
   (e) master theorem is not applicable;
   (f) none of the above.
4. Select correct (truthful) statements about dynamic programming:
   (a) Memoization is a dynamic programming technique, where we write the algorithm recursively in a natural manner, but modified to save the result of each subproblem.
   (b) Dynamic programming is impossible without dynamic arrays.
   (c) Top-down approach to dynamic programming generally computes more subproblems compared to bottom-up approach.
   (d) Dynamic programming is often used for optimization problems.

2 Question P (20 points)

2.1 Hashtable

The following array (indexed from 0) represents the internal array of integer keys for the hashtable («–» means free cell):

28	–	21	–	23	12	11	–	–	4	29	40

The hashtable uses **linear probing** to handle hash collisions, and uses the following hash function

$$h(k) = (k + 5) \mod 12$$

Determine a valid order in which the keys could have been added to the hashtable (assuming there were no removals, only inserts). If many orders are possible, specify any one.

2.2 Hashing

Answer the following questions about hashing:

1. True or False? Separate chaining requires more space asymptotically compared to open addressing approach.
2. True or False? When the load factor of the hash table is less than 1, the expected time complexity of core operations on the hash table with a uniform hash function is $\Theta(1)$.
3. Consider a hashing function that uniformly distributes keys in the N cells of the hashtable array. To store k entries, the expected number of keys in a bucket would be (select all correct answers):
   (a) k/N
   (b) N/k
   (c) $\Theta(1)$ assuming $k = \Theta(N)$
   (d) $\log N$
4. We consider a hash table of size $2n$ in which we insert $\frac{n}{8}$ keys, with a uniform hash function uniform. The expected (average) number of keys that have the same fixed hash value i is
   (a) $1/16$
   (b) $1/4$
   (c) $n/4$
   (d) $1/2$
5. Select all correct (truthful) statements about open addressing:
   (a) Linear probing tends to cluster entries of a hashtable into contiguous regions which can slow down `search` operation.
   (b) `search` operation is asymptotically faster with double hashing compared to linear and quadratic probing.
   (c) When the load factor reaches a certain threshold, the hash table can be rehashed into a new table of larger size in constant time.

3 Question S (25 points)

Edit distance (also known as *Levenshtein distance*) is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. The edit distance between two words is the minimum number of single-character edits (i.e., insertions, deletions, or substitutions) required to change one word into the other. Each of these operations has a unit cost. For example, the Levenshtein distance between `kitten` and `citing` is 4, since the minimal number of edit operation that transforms the former into the latter is 4:

1. `kitten` → `citten` substitute `k` with `c` at position 1
2. `citten` → `cittin` substitute `e` with `i` at position 5
3. `cittin` → `citting` insert `g` at position 7
4. `citting` → `citing` delete `t` at position 4

3.1 Finding Levenshtein distance

Consider the following two words: `structure` and `true`. Calculate the Levenshtein distance for these two words. Justify your answer by providing a series of single-character edits (insertions, deletions, or substitutions).

3.2 Brute forcing Levenshtein distance

Write down asymptotic time complexity of a Brute force algorithm for finding Levenshtein distance between two words of lengths n and m. Justify your answer, by explaining the idea in plain text (you do not have to to write pseudocode, or provide a formal proof).

3.3 Lists and Queues

Answer the following questions about lists and queues:

1. True or False? All operations of double-ended queue (deque) can be implemented in constant time (worst-case) when using doubly-linked lists to represent the queue.
2. Which data structure can be used to implement a queue?
   (a) Array
   (b) Resizable Array
   (c) Linked List
   (d) (a) and (b)
   (e) All of the above
3. Which data structure is most appropriate for efficiently searching a sorted list?
   (a) Array
   (b) Linked list
   (c) Stack
   (d) Queue
   (e) Hashtable

4 Question F (25 points)

4.1 Levenshtein distance with Dynamic Programming

Review definition of the Levenshtein distance in Question S. Describe a general algorithm solving the problem of finding the minimum edit distance using Dynamic Programming strategy for any two words of length n and m.:

1. Summarize the idea for a recursive algorithm.
2. Identify overlapping subproblems.
3. Write down pseudocode for the dynamic programming algorithm that solves the problem (top-down or bottom-up).
4. Provide asymptotic worst-case time complexity of the algorithm. Justify your answer.