CeTu - Initial technical assessment
C++ Software engineer

Overview:

The following document contains an initial technical assessment, which is the first technical step in the CeTu recruitment process. The aim of the assessment is to make sure we have basic understanding of the level and requirements for the role.

You are expected to answer the questions based on your knowledge. The parts that require implementation should be implemented by you. You can use any IDE of your choice to develop / test the attached code sample.

Note: The provided space for answers for each question is template based. Feel free to make the answer as long or as short as you see fit.

Hashmap

1. Describe what is the hashmap / hash table data structure.
   Give an example for when the structure is useful.

   Hashmap is a data structure. It stores keys and values of different types. The stored values are accessed by corresponding keys. No duplicates are allowed. To decrease the complexity of CRUD (create, update, delete) operations on keys and values certain methods are used. The hashing funtion (also called a digest) is used. The hashing function takes the key (the key may be of variable length) and generates a unique (to some degree) result that is used for fast value access. For example, we can calculate the remainder of the key using integer division – different hashing functions are used to reduce the amount of collisions. The hash collision is a situation when different keys result in the same hash, thus, reducing the access speed. Different hashing (key --> value mapping) functions have been developed, md5 (message digest version 5), for example. The hashing functions are studied and optimized using various mathematical methods, for example, numeric methods and statistics. As the keys are of variable length and the hashes are of fixed length, the hash collisions are inevitable. The concept of buckets is used – different keys with the same hash (with the keys themselves) are stored in a vector or a linked list, pointing to the values. Instead of direct access the linear search may be used. Usage example - a dictionary.

2. The following API is defined for a hash map.
   Implement the given API and address the constraints of the structure
   Note: You can't use std structures, use only classes / structures that you implement and new / delete constructs. I,e you are not allowed to use `std::map` / `std::unordered_map` or any other structure from the std library.

```
template<typename K, typename V>
class CeTuHashMap {
public:
    CeTuHashMap() {}

    // Insert a new pair into the hashmap
    void insert(K key, V value) {}

    // Lookup the given key in the map, if the key is not found return nullptr
    std::optional<V> lookup(K key) { }

    // Delete a pair with the key in the hashmap
    void erase(K key) {}
};
```

Explain what is the complexity of each method (e.g O(1), O(n), etc).

Complexity explanation:

void insert(K key, V value) {} : close to O(1), depends on a lot of factors and the implementation – the compiler, the compiler options, the memory management algotihms, the operating system used, the hardware (different RAM types, different latencies, BIOS configuartion), the temperature. The thing might even fail completely in the case of single upset event – a Solar wind particle might hit the RAM capasitor, flip the bit, and everything fails. Sometimes the backup systems are used, so that the probability of single upset event is reduced to the desired minimum, in case of going to Mars, for example.

std::optional<V> lookup(K key) { } : close to O(1), depends on a lot of factors and the implementation – the compiler, the compiler options, the memory management algotihms, the operating system used, the hardware (different RAM types, different latencies, BIOS configuartion), the temperature. The thing might even fail completely in the case of single upset event – a Solar wind particle might hit the RAM capasitor, flip the bit, and everything fails. Sometimes the backup systems are used, so that the probability of single upset event is reduced to the desired minimum, in case of going to Mars, for example.

void erase(K key) {} : close to O(1), depends on a lot of factors and the implementation – the compiler, the compiler options, the memory management algotihms, the operating system used, the hardware (different RAM types, different latencies, BIOS configuartion), the temperature. The thing might even fail completely in the case of single upset event – a Solar wind particle might hit the RAM capasitor, flip the bit, and everything fails. Sometimes the backup systems are used, so that the probability of single upset event is reduced to the desired minimum, in case of going to Mars, for example.

Use the following main function to test your implementation

```
int main() {
    // Test with int as both key and value
    CeTuHashMap<int, int> intMap;
    intMap.insert(1, 2);
    auto data = intMap.lookup(1);
    if (data) {
        cout << "data: " << *data << endl;
    } else {
        cout << "Key not found." << endl;
    }
    // Attempt to lookup a key that doesn't exist
    auto missingData = intMap.lookup(3);
    if (missingData) {
        cout << "Missing data: " << *missingData << endl;
    } else {
        cout << "Key 3 not found." << endl;
    }

    // Erase a key and then attempt to look it up
    intMap.erase(1);
    auto erasedData = intMap.lookup(1);
    if (erasedData) {
        cout << "Erased data: " << *erasedData << endl;
    } else {
        cout << "Key 1 not found after erase." << endl;
    }

    // Test with std::string as key and double as value
    CeTuHashMap<string, double> stringMap;
    stringMap.insert("pi", 3.14159);
    auto piValue = stringMap.lookup("pi");
    if (piValue) {
        cout << "pi: " << *piValue << endl;
    } else {
        cout << "Key 'pi' not found." << endl;
    }

    // Insert additional values and demonstrate lookup
    stringMap.insert("e", 2.71828);
    auto eValue = stringMap.lookup("e");
    if (eValue) {
        cout << "e: " << *eValue << endl;
    } else {
        cout << "Key 'e' not found." << endl;
    }

    // Erase a key and then attempt to look it up
    stringMap.erase("pi");
    auto erasedPiValue = stringMap.lookup("pi");
    if (erasedPiValue) {
        cout << "Erased pi value: " << *erasedPiValue << endl;
    } else {
        cout << "Key 'pi' not found after erase." << endl;
    }

    return 0;
}
```

Past Projects

Please describe one major project you had, what issues did you encounter, how did you approach solving those, and how did you analyze the issue. You can obfuscate parts that might be covered by NDA, and refer to different names or terms.

The BAMS project.

Developed the software for the drone interceptor rocket (everything from the bootloader to the target tracking system) and the ground software (everything from the protocols design to the logs and telemetry output) with a help of my CTO and colleagues.

Issues:

Had to educate myself with the help of my CTO in various domains: STM32F4 MCU and other microelectronic devices programming, some analog and digital circuit design and debugging. The team had no project manager, no rigid organizational structure, very scarce resources. I had my salary posponed several times, used my own resources while working for food and waiting for salary. One major issue was the deficit of planning, simulation and mathematical modelling. Another major issue was with the main control loop. I insisted on modifying it so that the target can be hit. The solution worked and the team managed to get the next round of investment. During the project I’ve read a very interesting book
called “Rockets and People” by Boris Chertok – I recommend you to read the book because it contains valuable information about rockets and people.