CSE 208: Data Structures and Algorithms-II Sessional
Offline 7: Hashing

Abrar Zahin Raihan
ID: 2105009
Section: A1

Report

Seed (for random sequence generator): $SEED = 69$
Maximum chain length of chaining: $l_{max} = 3$
Custom constants (for Custom Hashing): $C_1 = 37$, $C_2 = 41$
Maximum load factor (for double hashing and for custom hashing): $\lambda = 0.6$

Hash Table Size	**Collision Resolution Method**	**Hash1**		**Hash2**	
		# of Coll.	**Avg. Probes**	**# of Coll.**	**Avg. Probes**
5000	Chaining	879	1.02364	351	1.02531
	Double Hashing	3907	1.38000	3909	1.32000
	Custom Probing	3916	1.26000	3857	1.20000
10000	Chaining	820	1.03030	279	1.00753
	Double Hashing	3443	1.42000	3452	1.55000
	Custom Probing	3452	1.39000	3490	1.32000
20000	Chaining	713	1.03309	229	1.02047
	Double Hashing	2560	1.61000	2545	1.49500
	Custom Probing	2539	1.35500	2539	1.38500

Table 1: Hashing Method Comparison

Hash1()

```
int hash1(const std::string& str) {
    unsigned long hash = 5381; // Start with a large prime number
    for (char c : str) {
        // hash * 33 + c
        hash = ((hash << 5) + hash) + static_cast<unsigned char>(c);
    }
    // Ensure result is positive and fits in an int
    return (static_cast<int>(hash % INT_MAX)) % tableSize;
}
```

Listing 1: Hash function hash1()

Hash2()

```cpp
int hash2(const std::string& str) {
    unsigned long hash = 0; // Start with 0
    for (char c : str) {
        hash = static_cast<unsigned char>(c) + (hash << 6) + (hash << 16) - hash; // Similar
             to hash * 65599
    }
    return (static_cast<int>(hash % INT_MAX))%tableSize; // Ensure result is positive and
        fits in an int
}
```

Listing 2: Hash function hash2()

auxHash()

```cpp
    int auxHash(string key)
    {
        int ahash = 0;
        for (char c : key)
            ahash = ((ahash + c - 'a') * auxHashBase);
        ahash = table.size() - ahash % table.size();
        return ahash;
    }
```

Listing 3: Hash function auxHash()