Homework 5 : Answers

I. Problem Set

1. Key set : { 12, 9, 3, 0, 42, 98, 70, 1}

Separate Chaining Buckets :

hash(key) = (key * key+ 3) % 11

0	1	2	3	4	5	6	7	8	9	10
	3		0	12 ↓ 98 ↓ 1			9 ↓ 42	70		

All collisions successfully resolved.

Linear Probing

probe(i’) = (i + 1) % TableSize

0	1	2	3	4	5	6	7	8	9	10
	3		0	12	98	1	9	42	70	

All collisions successfully resolved.

Quadratic Probing

probe(i’) = (i*i + 5) % TableSize

0	1	2	3	4	5	6	7	8	9	10
	3	1	0	12	42		9	70	98	

All collisions successfully resolved.

Calculating the load factor :

λ = N / M
where N is the number of elements and M is the size of the hash table
λ = N / M
λ = 53491/100001
λ = 0.5349

2.

Function	Big O Complexity
Insert(x)	O(1)
Rehash ()	O(N)
Remove(x)	O(1)
Contains(x)	O(1)

3.

```
int hashit ( int key, int  tablesize)
{

        return ((key  * key  + 4) % tablesize);

}

int hashit( std :: string key, int tablesize)
{
        int size = key.length();
        int hashcode = 0;
        for(int i = 0; i < size; i++)
        {
                hashcode += (int) key[i];
        }
        return ((hashcode  * hashcode  + 4) % tablesize);
}
```

4. Parallel Programming :

   Parallel programming allows us to split a problem into small tasks and to run them simultaneously using multiple computer resources. In serial computing tasks are broken into a series of instructions and then executed one by one. Since only one instruction is executed at a moment resources are wasted. Parallel computing overcomes this problem.

5. Strategies for partitioning in parallel programming :

   The two major strategies for parallel programming are :

   a. <u>Task Parallelism</u> - Various tasks used in solving the problem are divided or partitioned among the cores.

   b. <u>Data Parallelism</u> -The data used is divided or partitioned among the cores. Each core carries out similar operations on it’s part of the data.