1.

Separate Chaining (Buckets)

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

Linear Probing

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

Quadratic Probing

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

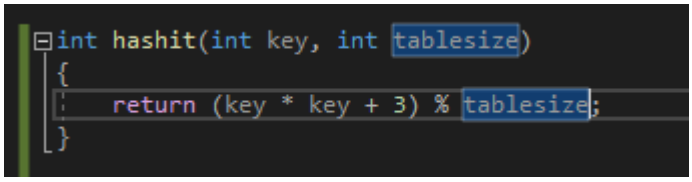
You will first need to decide if you need to rehash. When you insert a new item into the table you will bring up the data count to 0.53491 entries since the vector is sized at 100001 buckets.

2.

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

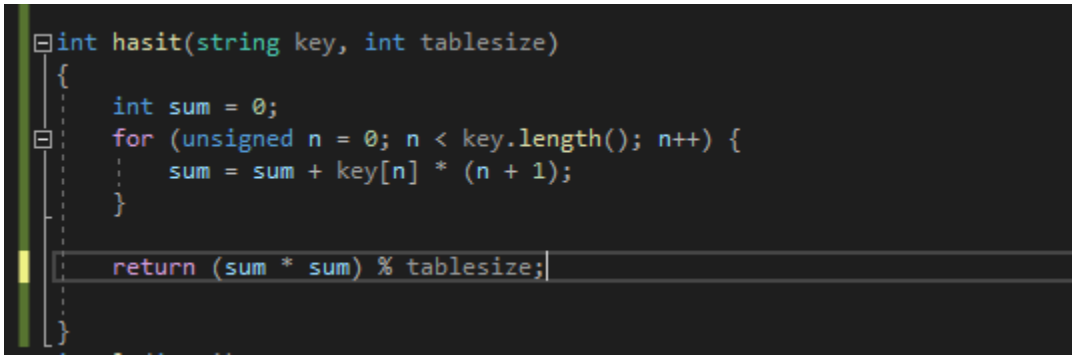
3.

a.



```
int hashit(int key, int tablesize)
{
    return (key * key + 3) % tablesize;
}
```

b.



```
int hasit(string key, int tablesize)
{
    int sum = 0;
    for (unsigned n = 0; n < key.length(); n++) {
        sum = sum + key[n] * (n + 1);
    }

    return (sum * sum) % tablesize;

}
```

4. Parallel programming is a way to divide up the workload of a program/s among the CPU cores to distribute the workload. The cores need to have good communication with each to send their partial sums, they need to have good load balancing to share work evenly and need synchronization to make sure one core does not get ahead of another.

5. One way to partition the workload is by **task parallelism**, which is partitioning various tasks use in solving the problem among the cores. The other way is by **data parallelism**, which is partitioning the data used in the problem among the cores and each core carries out similar operations on its part of the data.