Hashing - Exercise

Implement a hash table using the collision resolution techniques mentioned in the problems given below by implementing the following functions:

- function ‘create’ to create a hash table of size ‘size’ to store integer keys and return it
- function ‘insert’ to insert ‘key’ in the hash table using *h(x) = x mod size* as the hash function, and return the indices checked
- function ‘search’ to search for a ‘key’ in the hash table and return the index if found

Problem 1: Separate Chaining

Create a structure KeyNode as follows:

```
struct KeyNode{
        int key;
        struct KeyNode *next;
};
```

Function Prototypes:

- struct KeyNode * create(int size);
- int* insert(struct KeyNode *hashTable, int key);
- int search(struct KeyNode *hashTable, int key);

Problem 2: Quadratic Probing

Function Prototypes:

- int* create(int size);
- int* insert(int *hashTable, int key);
- int search(int *hashTable, int key);

Here, when ‘insert’ cannot find an empty slot to insert a key, a new hash table of double the size should be created and assigned to ‘hashTable’. All the keys in the old table are to be hashed into the new table and then the new key should be inserted into the new table. The old table can be freed.

Problem 3: Using Hash

Given an array of integers and a ‘sum’, implement a linear time algorithm to find the indices of 2 numbers in the array that add up to ‘sum’.

Function Prototype: int* twoSum(int array[], int sum);