Week 1

-Tanisha Gotadke (1BM21CS229)

Linear Probing

```
#include <stdio.h>
#include<stdlib.h>
#define SIZE 10

int h[SIZE]={NULL};

void insert()
{

 int key,index,i,flag=0,hkey;
 printf("enter a value to insert into hash table\n");
 scanf("%d",&key);
 hkey=key%SIZE;
 for(i=0;i<SIZE;i++)
    {

     index=(hkey+i)%SIZE;

     if(h[index] == NULL)
     {
        h[index]=key;
         break;
     }

    }

    if(i == SIZE)

     printf("\nelement cannot be inserted\n");
}
void search()
{

 int key,index,i,flag=0,hkey;
 printf("\nenter search element\n");
 scanf("%d",&key);
 hkey=key%TABLE_SIZE;
 for(i=0;i<TSIZE; i++)
 {
   index=(hkey+i)%SIZE;
   if(h[index]==key)
   {
     printf("value is found at index %d",index);
     break;
   }
  }
  if(i ==SIZE)
   printf("\n value is not found\n");
}
void display()
{

  int i;

  printf("\nelements in the hash table are \n");
  for(i=0;i<SIZE; i++)
  printf("\nat index %d \t value =  %d",i,h[i]);
}
main()
{
   int opt,i;
   while(1)
   {
       printf("\nPress 1. Insert\t 2. Display \t3. Search \t4.Exit \n");
       scanf("%d",&opt);
       switch(opt)
       {
           case 1:
               insert();
               break;
           case 2:
               display();
               break;
           case 3:
               search();
               break;
           case 4:exit(0);
       }
   }
}
```

Quadratic Probing

```
#include <stdio.h>
#include <stdbool.h>

#define SIZE 10int hash(int key, int attempt)
{
    return (key + attempt * attempt) % SIZE;}

void insert(int hashTable[], int key)
{
    int attempt = 0;
    while (attempt < SIZE) {
        int index = hash(key, attempt);
        if (hashTable[index] == -1) {
            hashTable[index] = key; // Insert the key at the index
            return;
        }
        attempt++;
    }
    printf("Hash table is full. Unable to insert %d.\n", key);
}

bool search(int hashTable[], int key)
{
    int attempt = 0; // Counter for quadratic probing attempts

    while (attempt < SIZE) {
        int index = hash(key, attempt); // Get the index for the key using quadratic hashing

        if (hashTable[index] == key) {
            return true; // Key found
        }
        attempt++;
    }

    return false; // Key not found
}

void display(int hashTable[])
{
    printf("Hash Table: ");
    for (int i = 0; i < SIZE; i++) {
        if (hashTable[i] != -1) {
            printf("%d ", hashTable[i]);
        } else {
            printf("_ ");
        }
    }
    printf("\n");
}

int main()
{
    int hashTable[SIZE];
    for (int i = 0; i < SIZE; i++) {
        hashTable[i] = -1;
    }
    int numKeys;
    printf("Enter the number of keys to insert: ");
    scanf("%d", &numKeys);

    printf("Enter the keys:\n");
    for (int i = 0; i < numKeys; i++) {
        int key;
        scanf("%d", &key);
        insert(hashTable, key);
    }

    display(hashTable);

    int searchKey;
    printf("Enter the key to search: ");
    scanf("%d", &searchKey);

    bool found = search(hashTable, searchKey);
    if (found) {
        printf("Key %d found in the hash table.\n", searchKey);
    } else {
        printf("Key %d not found in the hash table.\n", searchKey);
    }

    return 0;
}
```