Hashing (linear probing)

```
class Hash {
    int n, sum, tableSize, initialPos, temp;
    char enteredKey[30], enteredValue[30];

    struct data {
        char key[30];
        char value[30];
    } d[10];

public:
    Hash(int noOfEle) {
        n = noOfEle;
        int flag = 1; nearestPrime = n;
        while(flag) {
            flag = 0;
            nearestPrime++;
            for (int i=2; i<=n/2; i++) {
                if (nearestPrime % i == 0) {
                    flag = 1;
                }
            }
        }
        tableSize = nearestPrime;
        for(int i=0; i<tableSize; i++){
            strcpy(d[i].key, "\0");
            strcpy(d[i].value, "\0");
    } }
```

Teacher's Signature :

```
void insert(char key[], char value[]){ sum=0;
    for(int i=0; i<strlen(key);i++){
        sum+= ascii key[i];
    }
    initialPos= (sum+strlen(key))%tableSize;
    temp= initialPos;
    while(1){
        if(!strcmp(d[temp].key, "\0")){
            strcpy(d[temp].key, key);
            strcpy(d[temp].value, value);
            break;
        }
        temp= (temp+1)%tableSize;
        if(temp==initialPos){
            cout<<"cannot insert\n";
            break;
        }
    }
}

void delete(){
    cout<<"Enter key: ";
    cin>>key;
    sum=0;
```

```
for(int i=0; i< strlen(key) ; i++){
        sum+ = key[i];
}
initialPos = (sum + strlen(key)) % tableSize;
temp = initialPos;

while(1) {
    if (!strcmp(d[temp].key, key)){
        strcpy(d[temp].key, "\0");
        strcpy(d[temp].val, "\0");
        break;
    }
    temp=(temp+1) % tableSize;
    if(temp == initialPos) {
        cout<<" key not present \n";
        break;
    }
}
}

void search(){
    cout<<"Enter key: ";
    cin>>key;
    sum=0;
for(int i=0; i< strlen(key) ;i++){
        sum+=key[i];
}
```

```
initialPos = (sum/strlen(key)) % tableSize;
temp= initialPos;

while(1) {
    if (!strcmp(d[temp].key, key)){
      cout<<" value : " << d[temp].value;
      break;
    }
temp= (temp+1) % tableSize;
if (temp ==initialPos ){
   cout<< "Not Found";
   break;
  }
 }
}
```