1. Write a C program to implement hashing using Linear Probing method for {79,17,47,58,69,32,97}

Code:
```
#include <stdio.h>
#define SIZE 10

int hashTable[SIZE] = {0};

int hashCode(int key) {
   return key % SIZE;
}

void insert(int key) {
   int index = hashCode(key);
   while (hashTable[index] != 0) {
      index = (index + 1) % SIZE;
   }
   hashTable[index] = key;
}

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

int main() {
   int keys[] = {79, 17, 47, 58, 69, 32, 97};
   for (int i = 0; i < 7; i++) {
      insert(keys[i]);
   }
   display();
   return 0;
}
```

Output:
```
Hash Table: 58 69 32 97 0 0 0 17 47 79
```

2.**Write a C program to implement hashing using Separate hashing method for {10,12,23,42,53,62,74,85,96,105,116}**

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

struct node {
   int data;
   struct node *next;
};

struct node *hashTable[SIZE];

int hashCode(int key) {
   return key % SIZE;
}

void insert(int key) {
   int index = hashCode(key);
   struct node *newNode = (struct node *)malloc(sizeof(struct node));
   newNode->data = key;
   newNode->next = NULL;

   if (hashTable[index] == NULL) {
      hashTable[index] = newNode;
   } else {
      struct node *temp = hashTable[index];
      while (temp->next != NULL) {
         temp = temp->next;
      }
      temp->next = newNode;
   }
}

void display() {
   printf("Hash Table:\n");
```

```
    for (int i = 0; i < SIZE; i++) {
        printf("%d: ", i);
        struct node *temp = hashTable[i];
        while (temp != NULL) {
            printf("%d -> ", temp->data);
            temp = temp->next;
        }
        printf("NULL\n");
    }
}

int main() {
    for (int i = 0; i < SIZE; i++) {
        hashTable[i] = NULL;
    }

    int keys[] = {10, 12, 23, 42, 53, 62, 74, 85, 96, 105, 116};
    for (int i = 0; i < 11; i++) {
        insert(keys[i]);
    }
    display();

    return 0;
}
```

Output:

```
Hash Table:
0: 10 -> NULL
1: NULL
2: 12 -> 42 -> 62 -> NULL
3: 23 -> 53 -> NULL
4: 74 -> NULL
5: 85 -> 105 -> NULL
6: 96 -> 116 -> NULL
7: NULL
8: NULL
9: NULL
```