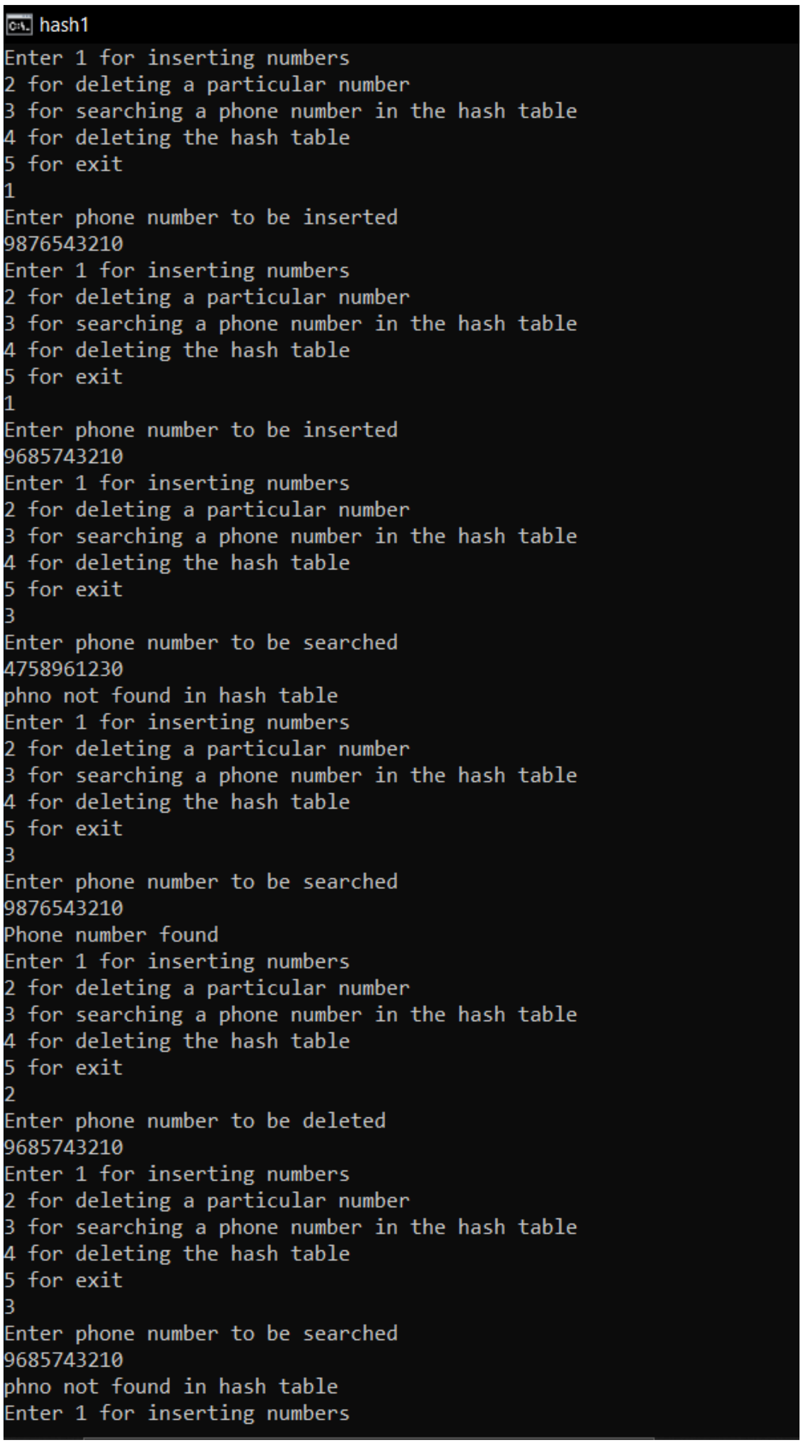
Problem : Collision using chaining

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 100
typedef struct node {
  long int phno;
  struct node *next;
} Node;
typedef struct hash {
  struct node *head;
  int count;
} HashTable;
HashTable *init()
{
  HashTable *temp = calloc(SIZE, sizeof(HashTable));
  for (int i = 0; i < SIZE; ++i)
  {
    temp[i].head = NULL;
    temp[i].count = 0;
  }
  return temp;
}
void destroy_hash(HashTable *hashtable)
{
  Node *temp = NULL, *to_del = NULL;
  for (int i = 0; i < SIZE; ++i)
  {
    temp = hashtable[i].head;
    hashtable[i].head = NULL;
    for (int j = 0; j < hashtable[i].count; ++j)
    {
      to_del = temp;
      temp = temp->next;
      free(to_del);
    }
  }
  free(hashtable);
}
void insert(HashTable *hashtable,long int phno)
{
  int hash = phno % SIZE;
  Node *new_node = (Node *)malloc(sizeof(Node));
  new_node->phno = phno;
  new_node->next = hashtable[hash].head;
  hashtable[hash].head = new_node;
  ++hashtable[hash].count;
}
void delete(HashTable *hashtable,long int phno)
{
  int hash = phno % SIZE;
  Node *temp = hashtable[hash].head, *prev = NULL;
  while (temp != NULL)
  {
    if (temp->phno == phno)
    {
      if (prev != NULL)
      {
        prev->next = temp->next;
        --hashtable[hash].count;
        free(temp);
        return;
      }
      else
      {
        hashtable[hash].head = temp->next;
        --hashtable[hash].count;
        free(temp);
        return;
      }
    }
    prev = temp;
    temp = temp->next;
  }
  printf("phno not found in hash table\n");
}
void search(HashTable *hashtable,long int phno)
{
  int hash = phno % SIZE;
  Node *temp = hashtable[hash].head;
  while (temp != NULL)
  {
    if (temp->phno == phno)
    {
      printf("%s\n","Phone number found");
      return;
    }
    temp=temp->next;
  }
  printf("phno not found in hash table\n");
}
int main()
{
  HashTable *p=init();
  int ch;
  long int num;
  while(1)
  {
    printf("Enter 1 for inserting numbers\n2 for deleting a particular number\n3 for searching a phone
number in the hash table\n4 for deleting the hash table\n5 for exit\n");
    scanf("%d",&ch);
    switch(ch)
    {
      case 1:
      printf("%s\n","Enter phone number to be inserted");
      scanf("%ld",&num);
      insert(p,num);
      break;
      case 2:
      printf("%s\n","Enter phone number to be deleted");
      scanf("%ld",&num);
      delete(p,num);
      break;
      case 3:
      printf("%s\n","Enter phone number to be searched");
      scanf("%ld",&num);
      search(p,num);
      break;
      case 4:
      destroy_hash(p);
      default:
      exit(0);
    }
  }
}
```



```
hash1
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
1
Enter phone number to be inserted
9876543210
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
1
Enter phone number to be inserted
9685743210
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
3
Enter phone number to be searched
4758961230
phno not found in hash table
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
3
Enter phone number to be searched
9876543210
Phone number found
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
2
Enter phone number to be deleted
9685743210
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
3
Enter phone number to be searched
9685743210
phno not found in hash table
Enter 1 for inserting numbers
```

Problem: Hashing using linear probing

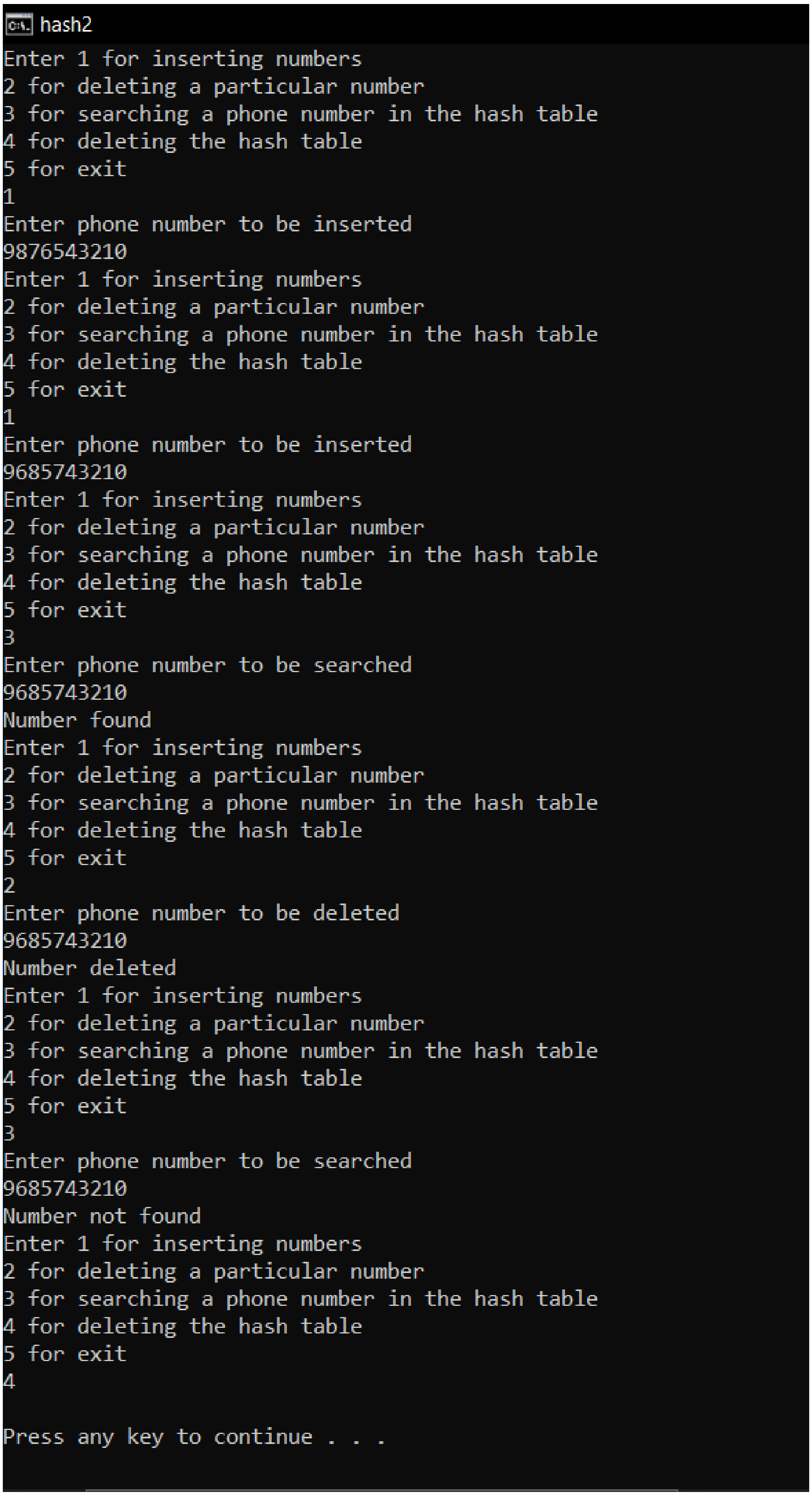
```
#include <stdio.h>

#include <stdlib.h>

#define SIZE 100

typedef struct {
 long int *table;
 int size;
} HashTable;
HashTable *init()
{
 HashTable *temp = malloc(sizeof(HashTable));
 temp->table = calloc(SIZE, sizeof(int));
 for (int i = 0; i < SIZE; ++i) {
 temp->table[i] = -1;
 }
 temp->size = SIZE;
 return temp;
}
void destroy_table(HashTable *htable)
{
 htable->size = 0;
 free(htable->table);
}
void insert(HashTable *htable,long int phno)
{
 int hash = phno % htable->size;
 int count = 0;
 while (count < htable->size)
 {
  if (htable->table[hash] == -1)
  {
   htable->table[hash] = phno;
   break;
  }
  ++hash;
  ++count;
```

```
    if (hash == htable->size)
    {
      hash = 0;
    }
  }
}
int search(HashTable *htable, long int phno)
{
  int hash = phno % htable->size;
  int count = 0;
  while (count < htable->size)
  {
    if (htable->table[hash] == phno)
    {
      return 1;
    }
    ++hash;
    ++count;
  }
  return 0;
}
void delete (HashTable *htable, long int phno)
{
  int hash = phno % htable->size;
  int count = 0;
  while (count < htable->size)
  {
    if (htable->table[hash] == phno)
    {
      htable->table[hash] = -1;
      printf("%s\n","Number deleted" );
      return;
    }
    ++hash;
    ++count;
  }
  printf("%s\n","Number not found" );
}
int main()
{
  HashTable *p=init();
  int ch;
  long int num;
  while(1)
  {
    printf("Enter 1 for inserting numbers\n2 for deleting a particular number\n3 for searching a phone
number in the hash table\n4 for deleting the hash table\n5 for exit\n");
    scanf("%d",&ch);
    switch(ch)
    {
      case 1:
      printf("%s\n","Enter phone number to be inserted");
      scanf("%ld",&num);
      insert(p,num);
      break;
      case 2:
      printf("%s\n","Enter phone number to be deleted");
      scanf("%ld",&num);
      delete(p,num);
      break;
      case 3:
      printf("%s\n","Enter phone number to be searched");
      scanf("%ld",&num);
      int j=search(p,num);
      if(j==1)
      printf("%s\n","Number found");
      else
      printf("%s\n","Number not found");
      break;
      case 4:
      destroy_table(p);
      default:
      exit(0);
     }
   }
 }
```



```
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
1
Enter phone number to be inserted
9876543210
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
1
Enter phone number to be inserted
9685743210
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
3
Enter phone number to be searched
9685743210
Number found
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
2
Enter phone number to be deleted
9685743210
Number deleted
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
3
Enter phone number to be searched
9685743210
Number not found
Enter 1 for inserting numbers
2 for deleting a particular number
3 for searching a phone number in the hash table
4 for deleting the hash table
5 for exit
4

Press any key to continue . . .
```