```
#include<stdio.h>

#include<stdlib.h>

#define LIMIT 30

enum record_status {EMPTY, DELETED, OCCUPIED};

struct Employee

{

int employee_id, employee_age;

char employee_name[30];

};

struct Record

{

struct Employee info;

enum record_status status;

};

int hash_function(int key)

{
```

```
return (key % LIMIT);

}

int search_records(int key, struct Record hash_table[])

{

int count, temp, position;

temp = hash_function(key);

position = temp;

for(count = 1; count != LIMIT - 1; count++)

{

if(hash_table[position].status == EMPTY)

{

return -1;

}

if(hash_table[position].info.employee_id == key)

{
```

```
return position;

}

position = (temp + count) % LIMIT;

}

return -1;

}

void insert_records(struct Employee emprec, struct Record hash_table[])

{

int count, position, temp;

int key = emprec.employee_id;

temp = hash_function(key);

position = temp;

for(count = 1; count != LIMIT - 1; count++)

{

if(hash_table[position].status == EMPTY || hash_table[position].status == DELETED)

{
```

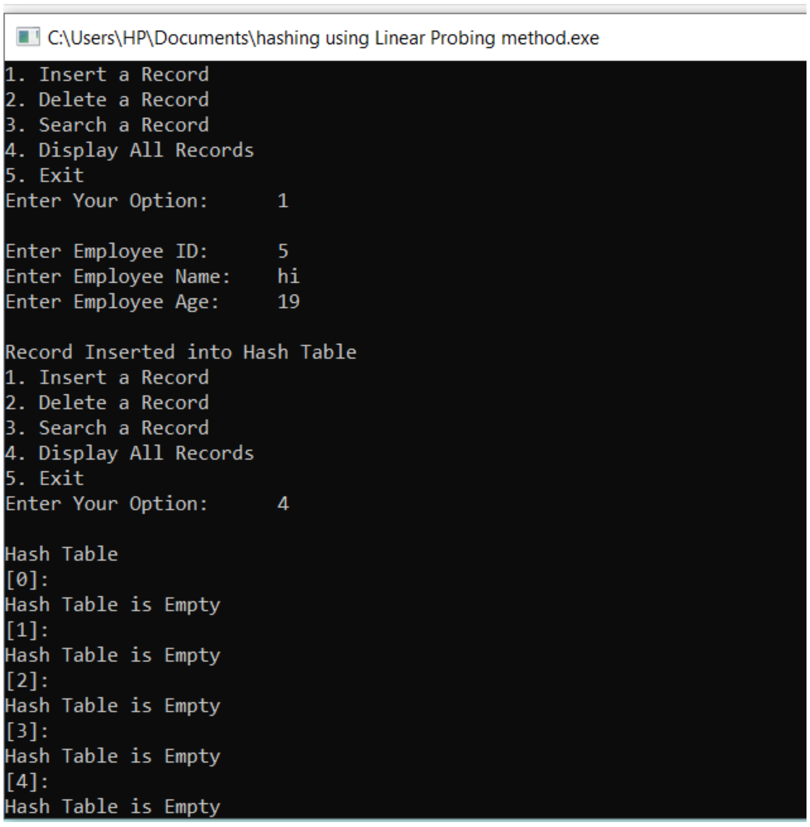
```
hash_table[position].info = emprec;

hash_table[position].status = OCCUPIED;

printf("\nRecord Inserted into Hash Table\n");

return;

}

if(hash_table[position].info.employee_id == key)

{

printf("\nDuplicate Record cannot be Inserted\n");

return;

}

position = (temp + count) % LIMIT;

}

printf("\nHash Table Limit Exceeded\n");

}

void display_records(struct Record hash_table[])
```

```
{

int count;

printf("\nHash Table\n");

for(count = 0; count < LIMIT; count++)

{

printf("[%d]:\t", count);

if(hash_table[count].status == OCCUPIED)

{

printf("Occupied - ID: %d Name: %s Age: %d",hash_table[count].info.employee_id,
hash_table[count].info.employee_name, hash_table[count].info.employee_age);

}

else if(hash_table[count].status == DELETED)

{

printf("\nRecord is Deleted\n");

}

else
{
printf("\nHash Table is Empty\n");
}
}
}
void delete_records(int key, struct Record hash_table[])
{
int position = search_records(key, hash_table);
if(position == -1)
{
printf("\nKey Not Found\n");
}
else
{
hash_table[position].status = DELETED;
}
```

```
}

int main()

{

int count, key, option;

struct Record hash_table[LIMIT];

struct Employee emprec;

for(count = 0; count <= LIMIT - 1; count++)

{

hash_table[count].status = EMPTY;

}

while(1)

{

printf("1. Insert a Record\n");

printf("2. Delete a Record\n");

printf("3. Search a Record\n");
printf("4. Display All Records\n");

printf("5. Exit\n");

printf("Enter Your Option:\t");

scanf("%d", &option);

switch(option)

{

case 1: printf("\nEnter Employee ID:\t");

scanf("%d", &emprec.employee_id);

printf("Enter Employee Name:\t");

scanf("%s", emprec.employee_name);

printf("Enter Employee Age:\t");

scanf("%d", &emprec.employee_age);

insert_records(emprec, hash_table);

break;

case 2: printf("\nEnter the Key to Delete:\t");

scanf("%d", &key);
```

```
delete_records(key, hash_table);

break;

case 3: printf("\nEnter the Key to Search:\t");

scanf("%d", &key);

count = search_records(key, hash_table);

if(count == -1)

{

printf("\nRecord Not Found\n");

}

else

{

printf("\nRecord Found at Index Position:\t%d\n", count);

}

break;

case 4: display_records(hash_table);
```

```
break;

case 5: exit(1);

}

}

return 0;

}
```



C:\Users\HP\Documents\hashing using Linear Probing method.exe

```
1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option:      1

Enter Employee ID:      5
Enter Employee Name:    hi
Enter Employee Age:     19

Record Inserted into Hash Table
1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option:      4

Hash Table
[0]:
Hash Table is Empty
[1]:
Hash Table is Empty
[2]:
Hash Table is Empty
[3]:
Hash Table is Empty
[4]:
Hash Table is Empty
```