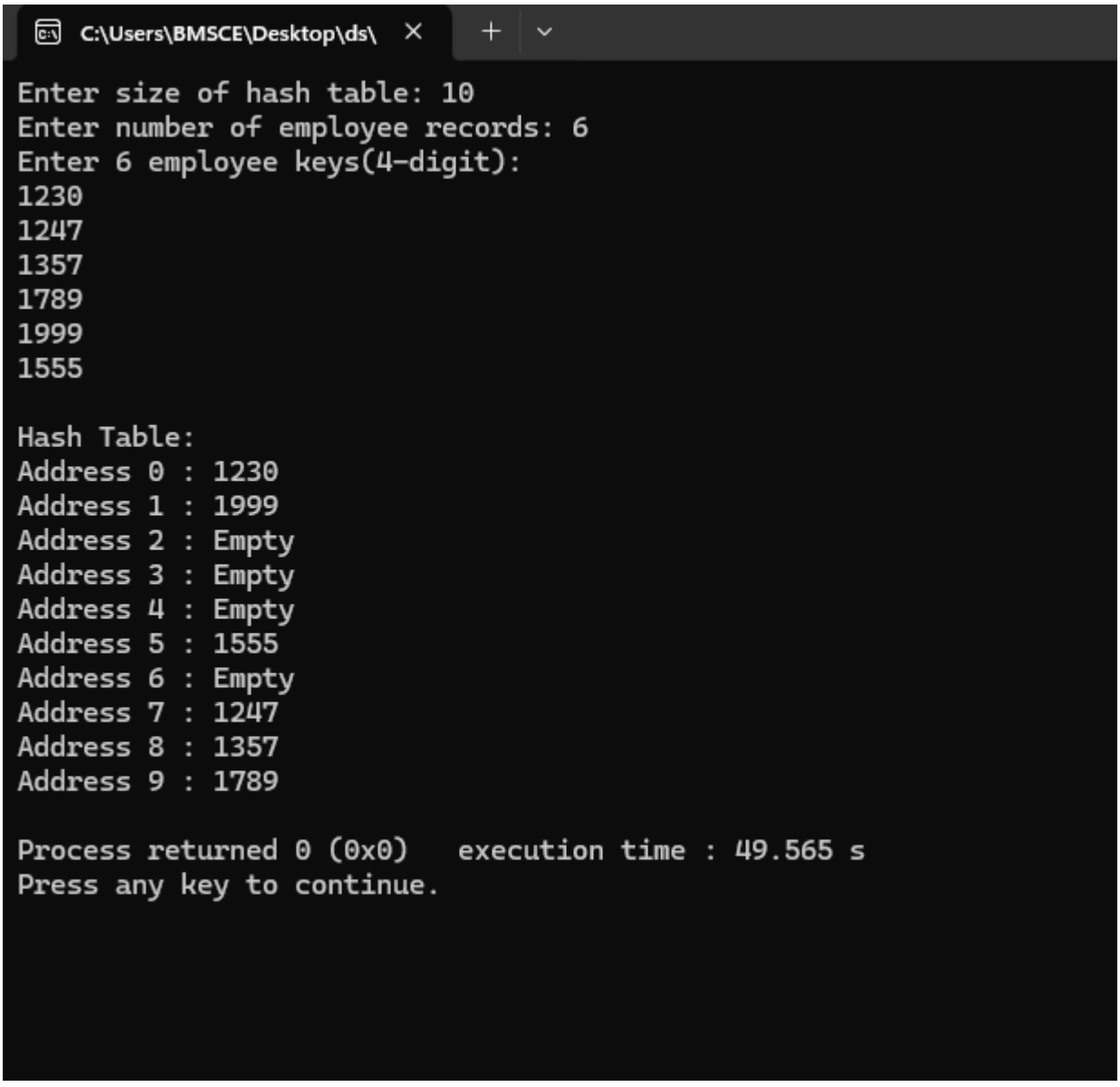
Prg 10:

```
#include <stdio.h>

#define MAX 20

int hashtable[MAX];
int m;

/* Function to insert key using Linear Probing */
void insert(int key)
{
   int index = key % m;

   if (hashtable[index] == -1)
   {
      hashtable[index] = key;
   }
   else
   {
      int i = 1;
      while (hashtable[(index + i) % m] != -1)
      {
         i++;
      }
      hashtable[(index + i) % m] = key;
   }
}
```

```
/* Function to display hash table */
void display()
{
   printf("\nHash Table:\n");
   for (int i = 0; i < m; i++)
   {
      if (hashtable[i] != -1)
         printf("Address %d : %d\n", i, hashtable[i]);
      else
         printf("Address %d : Empty\n", i);
   }
}

int main()
{
   int n, key;

   printf("Enter size of hash table: ");
   scanf("%d", &m);

   printf("Enter number of employee records: ");
   scanf("%d",&n);

   // Initialize hash table
   for (int i = 0; i < m; i++)
      hashtable[i] = -1;

   printf("Enter %d employee keys(4-digit):\n", n);
```

```
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &key);
        insert(key);
    }

    display();

    return 0;
}
```

OUTPUT:



OBSERVATION:

Page No:
Date.

22/10/25

Pg-10

Given a file of N employee records with a set K of keys (4-digit) which uniquely determine the records in file F.

Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are integers.

Design and develop a program in C that uses Hash Function 'H': K → L as H(K) = K mod m (remainder method) and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using Linear probing.

=>

```
# include <stdio.h>
# define MAX 20

int hashtable [MAX];
int m;

void insert (int key)
{
    int index = key % m;
    if (hash-table [index] == -1)
```

```
    }
    hashtable[index] = key;
  }
  else
  {
    int i = 1;
    while (hashtable[(index+i) % m] != -1)
    {
      i++;
    }
    hashtable[(index+i) % m] = key;
  }
}

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

int main() {
  int n, key;

  printf("Enter size of hash table: ");
```

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

printf("Enter number of employee records:");
scanf("%d", &n);

for(int i = 0; i<m; i++)
    hashtable[i] = -1;

printf("Enter %d employee keys (4-digit)
        :\n", n);
for(int i = 0; i<n; i++)
{
    scanf("%d", &key);
    insert(key);
}
display();

return 0;
}
```

Output :-

```
Enter size of hash table : 10
Enter number of employee records: 6
Enter 6 employee keys (4-digit):
1230
1247
1357
1789
1999
1555
```

Date.

Hash table :

Address 0 : 1030
Address 1 : 1999
Address 2 : Empty
Address 3 : Empty
Address 4 : Empty
Address 5 : 1555
Address 6 : Empty
Address 7 : 1047
Address 8 : 1357
Address 9 : 1789

22/12/25