Rajalakshmi Engineering College

Name: Manoj kumaar R
Email: 240801194@rajalakshmi.edu.in
Roll no: 240801194
Phone: 7708648574
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is building a basic hash table to manage student roll numbers for quick lookup. He decides to use Linear Probing to handle collisions.

Implement a hash table using linear probing where:

The hash function is: index = roll_number % table_sizeOn collision, check subsequent indexes (i+1, i+2, ...) until an empty slot is found.

You need to:

Insert a list of n student roll numbers into the hash table.Print the final state of the hash table. If a slot is empty, print -1.

Input Format

The first line of the input contains two integers n and table_size, where n is the

number of roll numbers to be inserted, and table_size is the size of the hash table.

The second line contains n space-separated integers — the roll numbers to insert into the hash table.

Output Format

The output should print a single line with table_size space-separated integers representing the final state of the hash table after all insertions.

If any slot remains unoccupied, it should be represented as -1.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4 7
50 700 76 85
Output: 700 50 85 -1 -1 -1 76

Answer

```
#include <stdio.h>

#define MAX 100

void initializeTable(int table[], int size) {
for (int i = 0; i < size; i++) {
table[i] = -1;
}
}
int linearProbe(int table[], int size, int num) {
int idx = num % size;
if (idx < 0) idx += size;
while (table[idx] != -1) {
idx = (idx + 1) % size;
}
return idx;
}
void insertIntoHashTable(int table[], int size, int arr[], int n) {
for (int i = 0; i < n; i++) {
```

```
int key = arr[i];
int idx = key % size;
if (idx < 0) idx += size;
if (table[idx] == -1) {
table[idx] = key;
} else {
table[ linearProbe(table, size, key) ] = key;
}
}
}
void printTable(int table[], int size) {
for (int i = 0; i < size; i++) {
printf("%d ", table[i]);
}
}
int main() {
    int n, table_size;
    scanf("%d %d", &n, &table_size);

    int arr[MAX];
    int table[MAX];

    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    initializeTable(table, table_size);
    insertIntoHashTable(table, table_size, arr, n);
    printTable(table, table_size);

    return 0;
}
```

Status : Correct **Marks : 10/10**