MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG315 Algorithms

Assignment 2: Student Attendance Application

This assignment aims to help you practice the **hash tables**, especially collision resolution with **open addressing** in hash tables. Your main task in this assignment is to develop a student attendance list for a seminar using **C programming language**.

Overview:
The office of international student affairs is hosting a seminar to acquaint newcomers with the rules and regulations and the different facilities METU NCC has to offer. However, due to Covid restrictions, the office intends to make a "signup to attend" application to know the number of attending students in advance so a suitable venue can be picked. In this assignment, you will develop said application.

You will first need to define a data structure to contain the student information in the table below.

Information	**Data Type**
Student ID (1 Letter followed by 2 digits), e.g. A34, E22, S21, etc.	char[3]
Name	char[40]
3-Letter Department Code, e.g. CNG	char[3]

The registration application shall be able to perform the following functionalities:

1. Add a student.
2. Search for a student using ID.
3. Print Table.

The application shall create a hash table in which the user decides what open addressing technique to be used: If the user enters (1) then the table will use **double hashing (where $f(i) = i * hash_2(x)$)**, but if they enter (2), then **quadratic probing (where $f(i) = i^2$)** will be used.

The initial size of the hash table should be 11. If the load factor λ (The total number of students in a hash table / the size of the hash table) becomes larger than 0.5, then the size of the hash table should be doubled and rounded to the next prime number, and then re-hashing should be performed.

The hash functions to be used:

- key = ASCII (studentid[0]) – 65 + studentid[1] + studentid[2]
- h_1(key) = (2 * key) mod hashtablesize
- h_2(key) = 7 – (key mod 7)

Process Model:
You can find the process model illustrated in Figure 1 below. The user will first pick whether **double hashing** or **quadratic probing** will be used to resolve collisions. After the choice is made, there is no going back on it unless the program is restarted. Then, the user will be presented with the following options:

1. **Add a student:** The operation will ask for all the necessary information. It will then create a student structure and place it in an appropriate position in the hash table. If the load factor of the hash table becomes more than 0.5, and then rehashing will be performed as mentioned above. An example is provided below where the input is shown in bold. If the id is not unique, then the application should print "Id should be unique!" and go back to the functionalities menu.

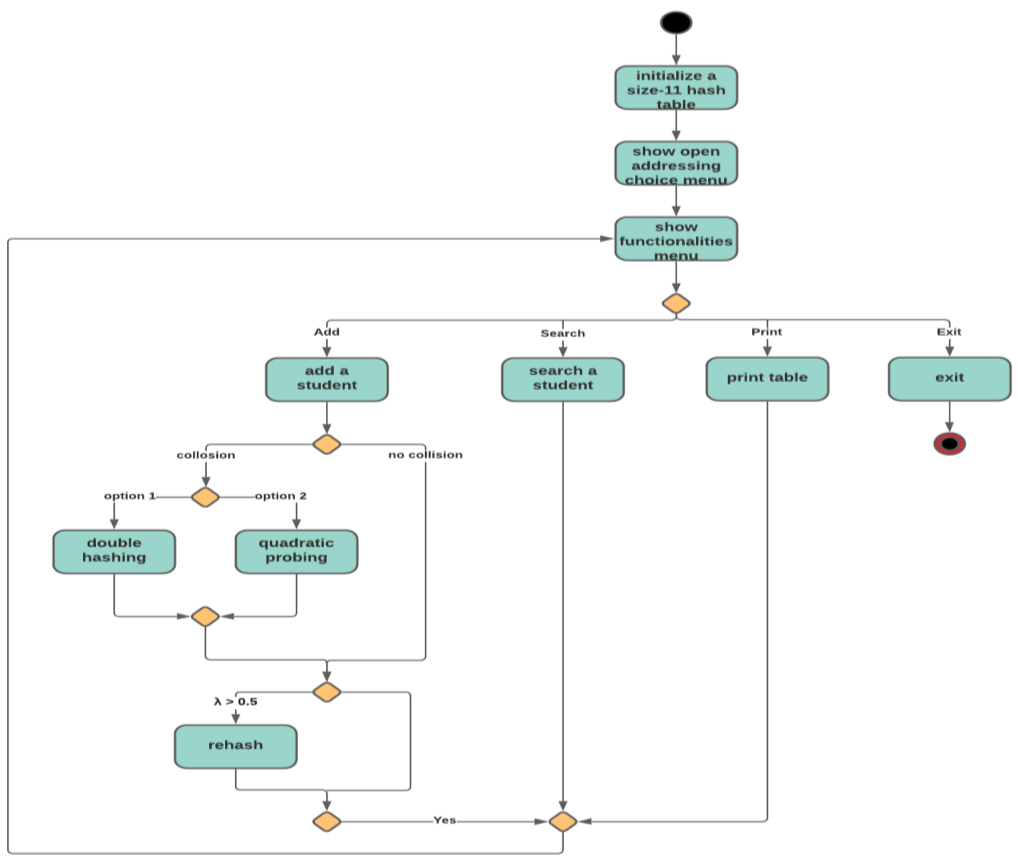

Figure 1 Process Model

```
Enter unique identifier: A69
Enter name: Peter Holland
Enter department: CNG
Peter Holland has been registered.
```

2. **Search a student:** The operation will ask for a student id and then try to find the student in the hash table and then print their details if found. An example is provided below where the input is shown in bold. If the student is not found, then the application should print "Student is not found!" and go back to the main menu.
```
Enter unique identifier: A69
Name: Peter Holland
Department: CNG
```

3. **Print Table:** This function will simply print the contents of the hash table in the order they are placed. As seen in the output below, if there is no element at a certain index, then the function will show nothing. You can find below examples of the output when either double hashing or quadratic probing are used. The input in the example is added in the order:
C12 Salley Bates CHM **->** A56 Ashley Norman MEC **->** A69 Peter Holland **->** A00 George Martin ASE **->** U11 Molly Eleanor PSY

Output with Double Hashing:
```
Index       ID          Name                    Department
0           A56         Ashley Norman           MEC
1
2
3
4
```

```
5
6           U11        Molly Eleanor         PSY
7           A00        George Martin         ASE
8           A69        Peter Holland         CNG
9
10          C12        Salley Bates          CHM
```

Output with Quadratic Probing:

```
Index       ID         Name                  Department
0           A56        Ashley Norman         MEC
1           A00        George Martin         ASE
2
3
4           U11        Molly Eleanor         PSY
5
6
7
8           A69        Peter Holland         CNG
9
10          C12        Salley Bates          CHM
```

Incremental and Modular Development

You are expected to follow an incremental development approach. Each time you complete a function or module, you are expected to test it to make sure that it works properly. You are also expected to follow modular programming which means that you are expected to divide your program into a set of meaningful functions.

Professionalism and Ethics

You are expected to complete the assignments on your own. Sharing your work with others, uploading the assignment to online websites to seek solutions, and/or presenting someone else's work as your own work will be considered as cheating.

Rules

- You need to write your program by using C programming.
- You need to name your file with your student id, e.g. 1234567.c, and submit to ODTUCLASS.
- Code quality, modularity, efficiency, maintainability, and appropriate comments will be part of the grading.
- **You need strictly follow the specifications given in this document.**

Grading Policy

The assignment will be graded as follows:

Grading Item	**Mark (out of 100)**
Student Data Structure	3
Main function where the menu operations are performed and appropriate functions are called.	15
Add student	13
Double hashing	12
Quadratic probing	12
Rehashing	19
Search student	16
Print table	10