MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG315 Algorithms

Date handed out: 29 November 2020, Sunday
Date submission due: 13 December 2020, Sunday 23:59

Assignment 1: A Simple Dog Shelter 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 simple dog shelter application using **C programming language**.

Overview

There are unfortunately many abandoned and lost animals. Animal shelters play an important role in helping these animals to survive and also finding them new homes. In this assignment, you will need to develop a simple dog shelter application. For each dog, the application should keep track of the following information. You will need to define a data structure to represent a dog. You will also need to define a data structure to represent a date.

Table 1. Dog Information

Information	Data type	When to set?
Unique identifier	Integer	When a dog is added
Name	Char array of Size 30	When a dog is added
Weight in kilograms	Float	When a dog is added
Height in meters	Float	When a dog is added
Entry date (day, month, year)	Date structure	When a dog is added
Leave date (day, month, year)	Date structure	When a dog is adopted

This simple dog shelter application should support the following functionalities:

- Add a dog to the shelter
- Search for a dog by using either its unique identifier
- Adopt a dog by using its unique identifier

The application should create a hash table using **open addressing with quadratic probing** (where f(i) = i^2) to keep track of these dogs where each element in a hash table represents a dog. The initial size of the hash table should be 11. If the load factor λ (The total number of dogs 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 function should be as follows where the key represents the unique identifier of a dog.

- `h(key) = key mod hashTableSize`

Process Model

The process model of this program is shown in Figure 1. As illustrated in the process model, the application will show a menu for the following operations:

1. Add a dog
   The operation will ask for all the necessary information and then create a dog structure. It then places it in an appropriate position in a 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 main menu.

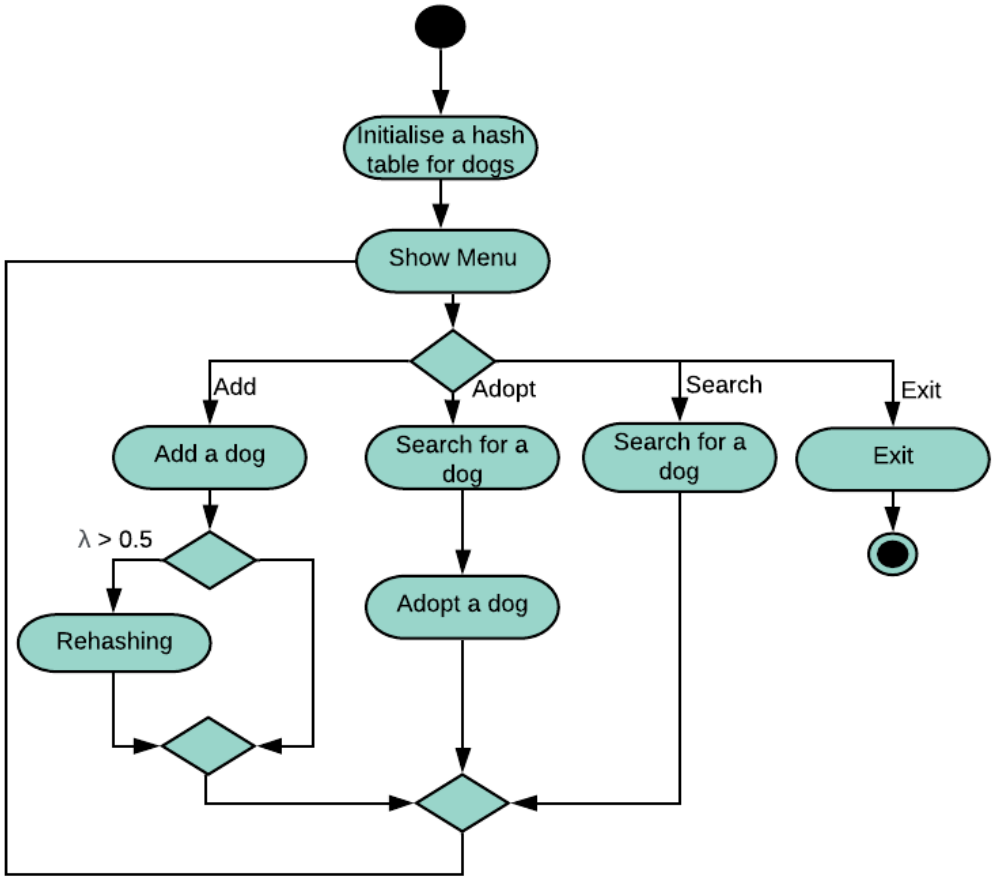

Figure 1. Process model

```
Enter unique identifier: 1
Enter name: Rex
Enter weight: 5.5
Enter height: 0.30
Enter entry date: 20.11.2020
Rex has been added to the dog shelter
```

2. Search for a dog
   The operation will ask for an identifier for a dog and then try to find the dog in the hash table and then print its details if found. An example is provided below where the input is shown in bold. The dog is not found, then the application should print "No dog is found" and go back to the main menu.

```
Enter unique identifier: 1
Name: Rex
Weight: 10
Height: 0.30
Entry Date: 20.11.2020
```

3. Adopt a dog
   The operation will ask for an identifier for a dog to be adopted and then the dog will be found in the hash table. The operation will then ask for a leave date to set the leave date of the dog. An example is provided below where the input is shown in bold. The dog is not found, then the application should print "No dog is found" and go back to the main menu. If the dog is already adopted, then the application should print "Dog is already adopted!" and go back to the main menu.

```
Enter unique identifier: 1
Name: Rex
Weight: 10
Height: 0.30
Entry Date: 20.11.2020
Enter leave date: 29.11.2020
Rex has been adopted.
```

4. Exit
   The operation will be used to terminate the application.

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 language.**
- You need to name your file with **your student id**, e.g. 1234567.c, and submit it to ODTUCLASS.
- **Code quality, modularity, efficiency, maintainability, and appropriate comments** will be part of the grading.

Grading Policy

The assignment will be graded as follows:

Grading Item	**Mark (out of 100)**
Data representation	15
Main program	10
Add a dog	20
Rehashing	25
Adopt a dog including search for a dog	20
Search for a dog	10