TASK 2 – Hash tables

Jade stores the details for her customers in a hash table. Each customer has a unique six-character code as a customer ID. Customer IDs start at AA0001 and range up to ZZ9999. A hashing function is used to create an index to store each customer. The hash table is implemented using a 1D array.

Key focus: Hashing functions

TASK 2.1

Discuss why a hashing function would be used to directly access a customer ID.

ANSWER

Customer ID's are unique values and we calculate the index from the key and assign indeed to the customer ID in the hash table. Calculating an index from a key is called 'hashing'. Finding a hashing function that will give a unique address from a unique key value is very difficult. If two different key values hash to the same address this is called a 'collision'.

TASK 2.2

Develop an algorithm to implement the hashing function. Make sure that every index of the array can be directly addressed

Write **pseudocode** for your algorithm.

ANSWER

```
DECLARE Customer: ARRAY[0:100000]OF STRING
DECLARE Index: INTEGER
FUNCTION Hash(customerID : STRING) RETURNS INTEGER
 Index ← ASCII( customerID[0])+ASCII(customerID[1]+INTEGER(customerID[2:5]))
 Customer[Index]←customerID
 RETURN Index
ENDFUNCTION
```

(or)

```
DECLARE Customer: ARRAY[0:100000]OF STRING
DECLARE Index: INTEGER
PROCEDURE Hash(customerID : STRING)
 Index ← ASCII( customerID[0])+ASCII(customerID[1]+INTEGER(customerID[2:5]))
 Customer[Index]←customerID
ENDPROCEDURE
```

TASK 2.3

Implement your algorithm in the programming language of your choice.

ANSWER

TASK 2.4

Expand your program to add 10 customer IDs to the hash table, to test your program. Your program should call the hashing function to do this.

ANSWER

TASK 2.5

Develop an algorithm to search the hash table for a customer ID.

Write **pseudocode** for your algorithm.

ANSWER

```
PROCEDURE Search(InputID: STRING)
 DECLARE Index : INTEGER
 Index ← ASCII(InputID[0]) + ASCII(InputID[1]) + INTEGER(InputID[2:5])
 IF Hash[Index] = InputID
     THEN
          OUTPUT "Found at array index", Index
     ELSE
          OUTPUT "Invalid"
     ENDIF
ENDPROCEDURE
```

TASK 2.6

Implement your search algorithm in **TASK 2.5** in the programming language of your choice.

ANSWER

TASK 2.7

Test your program using two of the previous customer IDs you added to the hash table.

Test your program using a customer ID that is not in the hash table.

ANSWER

TASK 2.8

It is possible that collisions could occur when using a hashing function.

Discuss what is meant by a collision, how they can occur and how they can be resolved.

ANSWER

Finding a hashing function that will give a unique address from a unique key value is very difficult. If two different key values hash to the same address this is called a 'collision'. Collisions can be resolved by creating an overflow area or storing the overflow record at the next available address in sequence. Re-design the hash function to generate a wider range of indexes // to create fewer collisions.

TASK 3 – Recursion

Key focus: Recursion

Recursion can be used when creating programs.

TASK 3.1

Discuss what is meant by the term recursion.

ANSWER

A function or procedure defined in terms of itself. Recursive solutions have a base case and a general case. The base case gives a result without involving the general case. The general case is defined in terms of the definition itself. It is very important that the general case must come closer to the base case with each recursion, for any starting point.

TASK 3.2

Consider the following pseudocode algorithm:

```
PROCEDURE NewYearCountdown(Value : INTEGER)
    IF Value = 0  ←  Base case
       THEN
          OUTPUT "Happy New Year"
       ELSE
          Value ← Value - 1  ←  General case
          OUTPUT Value
          NewYearCountdown(Value)  ←  Recursive call
    ENDIF
ENDPROCEDURE
```

In the algorithm, label the base case, the general case and the recursive call.

TASK 3.3

Recursive algorithms can be rewritten as an iterative loop.

Rewrite the recursive algorithm given in **TASK 3.2** as an iterative loop.

ANSWER

```
PROCEDURE NewYearCountdown(Value : INTEGER)

    WHILE Value>0

       THEN
          OUTPUT Value

          Value ← Value - 1
     ENDWHILE
     OUTPUT "Happy new year!"

ENDPROCEDURE
```

TASK 3.4

Using **pseudocode**, write a recursive algorithm that would output each letter in a string, from the beginning, one at a time. The string is passed as a parameter.

ANSWER

```
CALL NewYearCountdown(TestValue)
PROCEDURE Recursion(String : STRING) //String passed as a parameter
     IF LENGTH(String) = 0
          THEN
               OUTPUT "Completed"
          ELSE
               OUTPUT String[0]
               CALL Recursion(String[1:])
     ENDIF
ENDPROCEDURE
CALL Recursion("TestString")
```

TASK 3.5

Implement your algorithm in the programming language of your choice.

ANSWER