### LAB ASSIGNMENT

So we have reached the last session of basic programming lab practice. Thank you to have attend all programming activities with such anthusiasm. It is to our hope that the knowledge that you acquired in class and in the lab will build a strong informatics foundation for you to step forward in the world of Computer Science.

As our appreciation, in this module we will give you the key to succesfully facing the DAP final exam. The message containing the key can be acquired by completing the given **modul10.go** program template. Follow all the instructions.

Please remember that it is a confidential message, you can not share or discuss it, especiall with other who have not been able to crack the message.

**Good luck!**

**Directions**:

1. Copy the given program template into your text editor, and saved it as **module10.go**. Please compile it, using **go build modul10** to ascertain that there was no error in the program template.
2. Begin by typing down your name and sid at the allocate space (A).
3. Furnish the **readDict** procedure at location (C) so it will read the first two lines of input. Read the characters one by one as the spaces are also part of the input, i.e. use something similar to fmt.Printf("%c",&ch). Then the next line contains two integers, the **magic** and the **base** numbers.
4. Furnish the **readData** at location (D) to read pairs of data into the fields **thurs** and **dat** in the **dat** array. The list of pairs is ended a pair of marker **9999**. Store the number of pairs (except the marker 9999) into **ndat** parameter.
5. Furnish the **sortOnThurs** procedure at location (F) to sort the data using *selection sort* method. The sorted data should be in descending order (bigger to smaller) on **thurs** field.
6. Furnish the **sortOnDay** at location (G) to sort the **day** field using *insertion sort* method. The sorted data should be in descending order on **day** field.
7. Furnish the **keyOne** function at location (H) with *binary search* method on **thurs** field to find the location of the key to extract the data. The array is already in descending order on **thurs** fields. The **magic** value is guaranteed to be in the array somewhere.
8. Furnish the **keyTwo** function at location (J) by finding the median of the **days** fields. The array is already in descending order on **days** fields.
9. Furnish the **printSecretOne** at location (K) by extracting the information for the first text. The text is in the **days** fields. The modulo of that fields by the given **key** will be the index to the characters stored in the array **dict**. You have to print those characters using instruction similar to fmt.Printf("%c", dict[...]).
10. Furnish the **printSecretTwo** at location (L) by extracting the information for the second text. The text is in the **thurs** fields. The modulo of the that fields by the given **key** will be the index to the characters stored in the array **dict**. You can print them in the way similar to the previous procedure.
11. When everything is ok, trye the input example. If the output is correct, try the second input set. Copy the output from the second input set to location (B) near the beginning of your source program.

**template program module10.go**

package main

import "fmt"

/\* (A)

Name:

Sid:

(B) The output for the second input example:

\*/

const MAXCHAR = 256

type dataType struct {

thurs, day int

}

func readDict(dict \*[MAXCHAR]byte, ndict \*int, magic \*int, base \*int) {

var ch byte

var i int

/\* (C) read characters one at a time, end marker is a '.'

store the chars to dict, and

the number of characters to ndict

then read two integers for magic and base \*/

}

func readData(dat \*[MAXCHAR]dataType, ndat \*int) {

var i, val1, val2 int

/\* (D) read pairs of integers, ended with a pair of 9999

store them into thurs and day fields of array dat

store the number of data in ndat \*/

}

func sortOnThurs(dat \*[MAXCHAR]dataType, ndat int) {

/\* (F) sort on thurs fields in descending order

using selection sort \*/

}

func sortOnDay(dat \*[MAXCHAR]dataType, ndat int) {

/\* (G) sort on day fields in descending order

using insertion sort \*/

}

func keyOne(dat [MAXCHAR]dataType, ndat int, magic int, base int) int {

var pos int

/\* (H) set pos to index where magic is found in thurs fields,

using binary search method.

The array dat is in descending order on thurs \*/

return dat[pos].day % base

}

func keyTwo(dat [MAXCHAR]dataType, ndat int, base int) int {

var pos int

/\* (J) set pos to median index,

the array dat is in descending order on day \*/

return dat[pos].day % base

}

func printSecretOne(dat [MAXCHAR]dataType, ndat int, dict [MAXCHAR]byte, ndict int, magic, key int) {

for i := 0; i < ndat; i++ {

if dat[i].thurs != magic {

/\* (K) print a character from array dict.

The index is modulo key of the day field in dat \*/

}

}

fmt.Println()

}

func printSecretTwo(dat [MAXCHAR]dataType, ndat int, dict [MAXCHAR]byte, ndict int, magic, key int) {

/\* L do the same as printSecretOne, except now on thurs fields\*/

}

func main() {

var dict [MAXCHAR]byte

var ndict int

var magic, base int

var dat [MAXCHAR]dataType

var ndat int

var key int

readDict(&dict, &ndict, &magic, &base)

readData(&dat, &ndat)

sortOnThurs(&dat, ndat)

key = keyOne(dat, ndat, magic, base)

printSecretOne(dat, ndat, dict, ndict, magic, key)

sortOnDay(&dat, ndat)

key = keyTwo(dat, ndat, base)

printSecretTwo(dat, ndat, dict, ndict, magic, key)

}

**Input example**

#abcd efgh.

50 25

86 53

23 26

41 37

50 42

54 69

9999 9999

**Output example**

bach

cafe

**2nd Input example**

#we promptly judged antique ivory buckles for the next better prize!.

6666 123

1551 5195

2036 4830

2735 1390

3856 4514

3804 3946

7074 5862

2803 6846

3376 7604

4906 2996

2152 5790

6089 5579

5432 4096

6743 2129

7214 3503

6182 1958

5126 1865

5991 972

4335 7039

3185 6447

1940 5487

1278 7459

1634 2530

3593 1740

4533 5300

4168 3591

1471 6161

2248 1552

6581 4170

2497 2641

5513 3793

5310 6280

2899 5013

1766 4915

3679 1066

2425 4724

4768 5994

6666 4155

3303 3189

2983 6355

5711 4366

2324 1179

5857 2231

6366 7214

3939 2305

1164 2787

4807 6651

6923 2888

6280 3878

1087 3267

9999 9999