Department of Software / Computer Engineering

SENG114 & CENG104 – Computer Programming II

Spring 2021 - 2022

Lab Guide #7 – Week 8

OBJECTIVE : Practice on Recursive functions
Instructor : Yusuf Evren AYKAÇ **Assistants** : Elif ŞANLIALP, Ahmet Esad TOP, Nisanur MÜHÜRDAROĞLU MERCİMEK

1. Some algorithms require nested recursion where the result of one function call is a parameter to another function call. For example Ackermann's function is defined as:

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

Write a **recursive** function **Ackermann** that takes two integer numbers as parameters, finds and returns the result according to the given rules above.

Write a C program that gets two integer numbers from the user and computes the result of **isAckermann(m, n)**. The value of **m** and **n** both have got to be non-negative values (x >= 0).

Project_name: LabGuide7_1
File_name: Question_1.cpp

Example Run#1:
```
Enter the value of m: 3
Enter the value of n: 2
The result is 29
```

Example Run#2:
```
Enter the value of m: 2
Enter the value of n: -4
The value of both m & n had to be positive values. Exiting.
```

2. A perfect number is a number which the sum of its divisors are equal to the number itself. For example, 6 is a perfect number since the sum of its divisors 3, 2, and 1 adds up 6.

- Write a **recursive** function that finds returns the sum of divisors of a non-negative integer number.
- Write a main function that gets a number from the user and decides whether the number is perfect or not using function in the above description and displays a message as in the example run.

HINT: The maximum divisor of a number may be half of it.

Project_name: LabGuide7_2
File_name: Question_2.cpp

Example Run#1:
```
Enter a number: 6
6 is a perfect number!
```

Example Run#2:
```
Enter a number: 12 is NOT a perfect number!
```

3. Create an employee structure consisting of ID, name, surname and salary.

- Write a **recursive** function **bubble** that sorts the employees' data according to their IDs in ascending order using the bubble sort algorithm.
- Write a **recursive** function **binarySearch** that searches an employee with a given **ID** using the binary search algorithm.
- Write a main program that will read employee information from a text file whose name is given by the user**,** and store the data in a structure array. The program will also sort the data by using the **bubble** function.
- After sorting by using the **binarySearch** function, the program will find and display information about the employees according to the IDs gathered from the user as input until -1 is entered. If an ID could not be found in the list, display an appropriate message.

Project_name: LabGuide7_3
File_name: Question_3.cpp

Example Run:

```
Enter the file name: emp.txt
File not found Enter again: employee.txt
Enter an employee id (-1 to stop): 6598

Search Result
*************
6598 ALI        KURT       1380.0

Enter an employee id (-1 to stop): 1122
ID not found!!!

Enter an employee id (-1 to stop): 4312

Search Result
*************
4312 MEHMET     DOGRU       990.0

Enter an employee id (-1 to stop): -1
```

Employee.txt

```
1254 AHMET TUTUNCU 1500
1232 ALPER KAYI 1350
3425 GAMZE ALTAN 1000
6235 EMRAH KORAY 1255
8456 HUSEYIN BURAK 1450
1354 ONUR YILMAZ 1380
2344 EDA KAYA 1550
2312 GULCIN SEVER 2600
3412 FURKAN ALP 1000
4312 MEHMET DOGRU 990
4454 ERSIN KARGIN 975
6578 ALP DOGA 650
7645 HATICE TASTAN 1450
6598 ALI KURT 1380
9845 BURCU DOGAN 870
7546 MERAL KURAL 1475
3499 BERK SAVAS 890
6583 DEFNE VURMAZ 1100
5349 TAMER COSKUN 1250
7087 VURAL KINAY 2480
```

4. Write a program that will read sorted and structured city listings (**city name**, **where the city stays on**, **population**) from a text file named as **cities.txt**, and store it into a binary file. Use below functions:

Write a function **readAndWrite** that reads all the city data from a text file, latter, writes the city data to a binary file. This function returns the count of the cities.

Write a **recursive** ***BinarySearch*** function that searches a given **city name** into the binary file by using the recursive binary search algorithm.

By using the above functions, the program will find and display specific city information according to the city name gathered from the user as input by using the function that you are going to write ***BinarySearch***. If the city could not be found in the list, display an appropriate message.

cities.txt

```
Ahmedabad India 2954526
Alexandria Egypt 3339076
Ankara Turkey 4984099
Baghdad Iraq 3841268
Bandung Indonesia 5919400
Bangalore India 3302296
Bangkok Thailand 7506700
Beijing China 7362426
Berlin Germany 3388000
Bogor Indonesia 5000100
Bogota Colombia 6422198
Bombay India 15925891
Cairo Egypt 6800992
Calcutta India 4399819
Chengdu China 2954872
Chicago UnitedStates 2896016
Chongqing China 3122704
Delhi India 7206704
Guangzhou China 3935193
Harbin China 2990921
Hyderabad India 3145939
Istanbul Turkey 10260438
Jakarta Indonesia 9373900
Karachi Pakistan 9339023
Lahore Pakistan 5143495
Lima Peru 6414500
Madras India 3841396
Madrid Spain 2823667
Melbourne Australia 3413894
MexicoCity Mexico 8235744
Moscow Russia 8297056
Santiago Chile 4788543
Shanghai China 8214384
Shenyang China 4669737
Singapore Singapore 3894000
Sydney Australia 4031944
Teheran Iran 6758845
Tianjin China 5855044
Tokyo Japan 8130408
Wuhan China 4040113
Xian China 2872539
Yokohama Japan 3426506
```

Example Run#1:

Enter a city name: Bombay
Bombay India 9925891

Example Run#2:

Enter a city name: Ankara
Ankara Turkey 2984099

Example Run#3:

Enter a city name: Konya
"Konya" could not be found!!!

Project_name: LabGuide7_4
File_name: Question_4.cpp