

Programme Name: **BCS hons**

Course Code: **CSC 1513**

Course Name: **PROGRAMING FUNDAMENTALS**

**Open Book Programming**

Date of Submission: **7/10/2020**

**Submitted By: Submitted To:**

Student Name: **Dipesh Tha Shrestha** Faculty Name: **PRAKASH CHANDRA**

IUKL ID: **041902900028**  Department: **LMS**

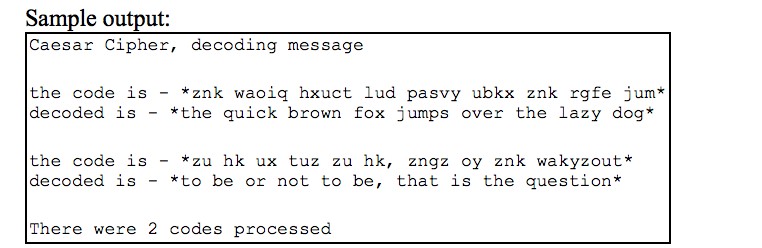
Semester: **Second**

Intake: **Sept 2019**

1. Write a program to decode Caesar Cipher secret messages, using the numeric key from provided dataSam5a.txt. Each line in the data file, dataSam5b.txt will be a string to be converted from the garbled characters into readable characters. The numeric key is the positive integer value that was added to the ascii character‟s numeric value to get the coded message. You should **subtract** that value to decode[remember to keep the character values in the range of „a‟-„z‟ have numeric value between 97(a) and 122(z), if a character is outside after the conversion, force that character back into the range]. Any characters that do not start in that range should just be passed straight through, for instance the space(character = 32). All letters will be lower-case, punctuation and spaces should be passed straight through.

The program should print the complete message as a single string. write the output to a file called *output1.txt* as well as to terminal.

The sample output is shown below:-



***Answer***

The Caesar Cipher technique is one of the earliest and simplest method of encryption technique. It’s simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter some fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.  
Thus to cipher a given text we need an integer value, known as shift which indicates the number of position each letter of the text has been moved down.  
The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| import java.util.\*;public class CaesarCipherProgram {        public static void main(String args[]) {        String ciphertext ="znk waoiq hxuct lud pasvy ubkx znk rgfe jum";        int shift = 6;        String decryptMessage = "";        for(int i=0; i < ciphertext.length();i++)        {            char alphabet = ciphertext.charAt(i);            if(alphabet >= 'a' && alphabet <= 'z')            {                alphabet = (char) (alphabet - shift);                if(alphabet < 'a') {                    alphabet = (char) (alphabet-'a'+'z'+1);                }                decryptMessage = decryptMessage + alphabet;            }            else if(alphabet >= 'A' && alphabet <= 'Z')            {                alphabet = (char) (alphabet - shift);                if (alphabet < 'A') {                    alphabet = (char) (alphabet-'A'+'Z'+1);                }                decryptMessage = decryptMessage + alphabet;            }            else            {             decryptMessage = decryptMessage + alphabet;            }        }        System.out.println(" decrypt message : " + decryptMessage);    }}code 2;import java.util.\*;public class CaesarCipherProgram {        public static void main(String args[]) {        String ciphertext ="zu hu ux tuz zu hk, zngz oy znk wakyzout";        int shift = 6;        String decryptMessage = "";        for(int i=0; i < ciphertext.length();i++)        {            char alphabet = ciphertext.charAt(i);            if(alphabet >= 'a' && alphabet <= 'z')            {                alphabet = (char) (alphabet - shift);                if(alphabet < 'a') {                    alphabet = (char) (alphabet-'a'+'z'+1);                }                decryptMessage = decryptMessage + alphabet;            }            else if(alphabet >= 'A' && alphabet <= 'Z')            {                alphabet = (char) (alphabet - shift);                if (alphabet < 'A') {                    alphabet = (char) (alphabet-'A'+'Z'+1);                }                decryptMessage = decryptMessage + alphabet;            }            else            {             decryptMessage = decryptMessage + alphabet;            }        }        System.out.println(" decrypt message : " + decryptMessage);    }}  |  |  | | --- | --- | |  |  | |  |  |

1. Write a program to read in a set of **real** scores(no more than **150** scores) from provided **data7.txt data file**  and count the scores, print the scores out 7 per line, calculate the mean (average) of the numbers. **You should print to a file called *output2.txt*.**  You must use a method to read the data into the array until Java look ahead EOF is detected. **NOTE: use the data7.txt data file**

You **MUST** write several methods for this question:

* 1. A method to print the arrays **ls** per line(where **ls** is an integer value of the line size or number of values printed per line), ls should be one of the parameters. Use this method for all output of the array. This function should leave the output environment ready to start at the beginning of a newLine. Use the following header.

*public static void prntArray(PrintWriter fl,double a[],int num,int ls) /\* ls is line size(number per line) \*/*

* 1. A method to read real values into the array until out of data, returning the integer number of scores read in (less than or equal to 150). Use one of the following headers.

*public static int readArray(Scanner fl,double x[]) //use arrayLength public static int readArray(Scanner fl,double x[],int num) // pass max size*

* 1. A method for the mean(average) calculation. Use the following header.

*public static double calcMean(double a[],int num)*

**The sample output is as follows:**

|  |
| --- |
| 3.00 3.70 2.00 5.30 1.00 7.30 3.00  3.60 7.00 11.10 2.00 6.41 3.00 6.20  5.00 3.70 2.00 8.65 7.00 10.50 5.00  7.20 4.00 13.75 2.00 5.30 1.00 7.45  7.00 4.35 2.00 4.60 5.00 10.40 7.00  8.25 4.00 8.70 3.00 9.35 5.00 8.10  3.00 6.70 4.00 2.50 1.00 2.60 5.00  3.70 2.00 5.80 1.00 4.90    There are 54 numbers in array with mean of 5.11 |

***ANSWER***

# import java.io.File;

# import java.io.FileNotFoundException;

# import java.io.PrintWriter;

# import java.util.Scanner;

# public class Scores {

# static double [] array= new double[150];

# public static void main(String[] args) throws Exception{

# File file= new File("C:\\user\\Desktop \\data7.txt");

# Scanner fileinput = new Scanner(file);

# PrintWriter out = new PrintWriter("output2.txt");

# int number = readArray(fileinput,array);

# printArray( out,array,number,7);

# }

# public static int readArray(Scanner fl, double[] x) throws FileNotFoundException {

# int countfile=0;

# Scanner fileinput = new Scanner(new File("src/com/company/data7.txt"));

# while(fileinput.hasNext()){

# String line = fileinput.nextLine();

# countfile++;

# }

# for (int i = 0 ; i< countfile; i ++){

# x[i]= Double.parseDouble(fl.nextLine());

# }

# array= x;

# return countfile;

# }

# public static void printArray(PrintWriter fl, double[] a, int num, int ls){

# for ( int i = 0; i < num;i++ ){

# for (int j = i ; j< i + ls;j++ ){

# fl.write(a[j]+" ");

# }

# i+=ls-1;

# fl.write("\n");

# }

# fl.write("There are "+num+" numbers in array");

# double mean = calcMean(array,num);

# fl.write("\nwith a mean of "+ mean );

# fl.close();

# }

# public static double calcMean(double[] a, int num){

# double sum= 0 ;

# for ( int i = 0 ; i < num ; i ++){

# sum += a[i];

# }

# return sum/num;

# }

# }

1. A Stack is a data structure where you add elements to the "top" of the stack, and also remove elements from the top again. This is also referred to as the "Last in First out (LIFO)" principle. Write JAVA program to implement stack data structure as per the following class diagram:

|  |
| --- |
| Stack |
| Stck [ ]: int  tos: int |
| Stack(size:int)  Push(item:int): void  Pop(): void  Isempty(): boolean  Isfull(): boolean |

|  |  |  |
| --- | --- | --- |
|  | push(*new-item*: integer) | True if no more items can be pushed. If stack is not FULL then Adds an item onto display “STACK OVERFLOW”  If stack is not EMPTY then Removes the most-recently-pushed item from the stack otherwise display “STACK UNDERFLOW”  True if no more items can be popped and there is no top item. |
| the stack otherwise  pop(): Integer  is-empty(): Boolean  is-full(): Boolean |

NOTE: Initialize top of stack to -1, when stack is first created. Sample output:

***ANSWER***

# import java.util.Scanner;

# public class Stack{

# int[] stck;

# int tos;

# int maxsize;

# 

# 

# Stack(int size){

# tos = -1;

# this.maxsize = size;

# stck = new int[maxsize];

# 

# }

# 

# void Push(int item){

# if(tos ==maxsize-1){

# System.out.println("STACK OVERFLOW");

# }

# stck[++tos] = item;

# }

# 

# 

# int  Pop()

# {

# if (tos<0)

# {

# throw new RuntimeException("STACK UNDERFLOW");

# 

# }

# else {

# return stck[tos--];

# }

# 

# }

# 

# boolean Isempty(){

# return (tos<0);

# 

# }

# 

# boolean isfull(){

# return (tos == maxsize-1);

# }

# public static void main(String [] args)

# {

# Scanner sc = new Scanner(System.in);

# System.out.print("Enter the  size of stack:");

# int s = sc.nextInt();

# System.out.println("Enter "+s+" elements to stack:");

# Stack s1 = new Stack(s);

# for(int i =0;i<s;i++){

# int num = sc.nextInt();

# s1.Push(num);

# }

# System.out.println("Stack is full");

# for(int j = 0;j<s;j++)

# {

# System.out.println("Item deleted:"+s1.Pop());

# 

# }

# System.out.println("stack underflow");

# System.out.println("stack is empty");

# }

# 

# 

# 

# }