CS19003: Programming and Data Structures Laboratory
Autumn semester 2021-22, Section 16
Guidelines to Students on Malpractice

Any student is expected to write his/her code for assignments and tests completely by himself/herself, and not indulge in any malpractice or adopt any unfair means. You are also required to not share your code with anyone else, either directly or indirectly through other means such as sharing passwords of accounts containing your code. It is strongly emphasized that cheating in any form in assignments and tests is a serious offence and will be dealt with very strongly if detected.

Every student is requested to fully read and understand the following carefully. No excuses will be tolerated later.

1. A program submitted for an assignment/test will be considered as plagiarized (copied) if any of the following occurs:
   - Substantial similarity with submission of another student
   - Substantial similarity with any other external resource (copy from internet, book etc.)
   - If the student is not able to reasonably explain the code written by him or her if asked by the teacher

   The teacher will decide the similarity level based on the specific assignment/test and other factors, and the decision of the teacher will be final.

2. If a plagiarism is detected and confirmed, there will be a minimum penalty of assigning 0 to the submissions involved without distinguishing the role of a student involved in the event (whether the student copied or whether the student shared his/her code with another student). The teacher can impose any additional penalty as deemed fit by him/her depending on the nature of the offence, including referring to the Institute disciplinary committee. The policy will be announced by your section teacher at the beginning of the course.

   We will be very strict in case of any plagiarism and code copying. Take extreme precaution for protecting your account information, and for not sharing your programs that you would be submitting for evaluation.

3. If plagiarism is detected, no distinction will be made as to who copied from who. Note that if you allow someone to copy from you, you will be considered equally guilty as the person copying. No excuses will be tolerated in this regard.

4. You are requested to protect your code and not share it with anyone. You are also requested to protect passwords of your accounts in moodle etc. so that your code cannot be accessed by anyone else. If a copy is detected from your code, no excuses will be considered (Some common excuses we hear are “I have given him/her my code to see, I

didn’t know he/she will copy”, “I have given him/her my moodle password, I didn’t know he/she will copy my code”, “I do not how he/she copied my code, maybe they found my moodle password” etc.).

Sometimes the teacher will allow the use of some resources explicitly; as an example, while doing assignments, you are usually allowed to look up your textbook for help with C language syntax etc. **However, do not assume anything unless the teacher announces it explicitly**. I**f you have any confusion as to what you can and cannot do, always ask your teacher or TA first**, they are always there to help you. The most important thing in the class is for you to learn and help will always be available.

Department of Computer Science and Engineering

Assignment 1

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).

Answer all the questions. **[20 x 5 = 100 Marks]**

1. Write a C program to print on the screen the following lines in the exact same sequence as given. You cannot use more than TWO printf() to print everything on your screen.

   "I am learning C Programming. I am very excited.
   This is my first time." **[20]**

2. The office in which you are working has decided to contribute a part of your salary towards a special fund contribution. Your contribution is calculated as the addition of 10% of your basic pay and 5% of your Dearness Allowance (DA). Write a program to take input two integers as your basic pay and as your DA and display the fund contribution.

   Example:

   Input: Basic Pay: 1000

   DA : 200

   Output: Fund contribution: 110 (10% of 1000 + 5% of 200) **[20]**

3. Complex numbers are represented geometrically as a point in a 2D Argand plane, where the x-axis denotes the real part and the y-axis denotes the imaginary part. Take input one complex number from the keyboard. Hence *reflect* this complex number in the Argand plane with respect to the vertical y-axis.

   (a) Display the reflected complex number in the form "a + ib" (i.e if the real part is 2 and the imaginary part is -5 then you should display "2-i5" on the screen).

   (b) Additionally, display the area enclosed by the rectangle between the horizontal x-axis (the real axis) and the straight horizontal line connecting the two complex numbers, i.e the original and the reflected complex number. The other two sides of the rectangle are the vertical height of the imaginary part of the two complex numbers.

   [**Hint:** For input, maintain two separate variables, one denoting the real and the other imaginary part for one complex number. You can assume all integer variables.] **[20]**

4. Given the base radius and total surface area (curved surface area + top circular area + bottom circular area) of an enclosed right circular cylinder in cm, first calculate the height of the cylinder (in cm) and then find its volume (in cm^3). Hence, calculate the cost of storing a perfume to fill exactly $(2/3)^{rd}$ of the cylinder at an amount of Rs.10 per cubic centimeter and the remaining volume is filled with Nitrogen gas at Rs.2 per cubic centimeter. Your input should be the radius and surface area of the cylinder (in cm and cm^2 respectively) and you should display the height, volume and the cost of storing the perfume with the Nitrogen gas.

   Assume,

   $\pi = 3.141$

   Curved surface area of cylinder = 2*π*base radius*height

   Volume = π*radius^2*height

5. If a five-digit number is taken as input through the keyboard, write a C program to print the difference of the sum of the last three digits **from** the sum of the first two digits of the number. Also use **one** single integer variable to generate the reverse of the last three digits and print that single integer [**Hint:** Use positional notation for decimal numbers to generate the reverse number.].

   Example:

   Input:

   97321

   Output:

   Difference = 10 (sum of (9,7) - sum of (3,2,1))

   Reverse number = 123 (as a single integer) **[20]**

Department of Computer Science and Engineering

Assignment 2

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).
- If you do not follow the instructions, your marks may be deducted.

Answer all the questions. **[20 + 20 + 30 + 30 = 100 Marks]**

1. Read three real numbers, i.e., a, b, and c, which represent the coefficients of a quadratic equation and display them in the exact form $ax \hat{} 2 + bx + c = 0$. Hence check whether the roots are real, coincident, or complex according to the rules of quadratic equation and display it. No need to find out the roots separately. Additionally if the roots are complex, determine the quadrant (i.e 1^{st} quadrant/2^{nd} quadrant/3^{rd} quadrant/4^{th} quadrant) of the 2D Argand plane in which the first complex root of the form $a + ib$ lies. Note that complex roots occur as conjugate pairs $a+ib$ and $a-ib$, consider only the root in the form $a+ib$ and determine in which quadrant it lies.

   Example1:
   (Please take the input and display the output as shown)

   Input:

   a:5

   b:2

   c:1

   Output:

   Equation: $5x \hat{} 2 + 2x + 1 = 0$

   Roots: Complex

   Quadrant of 1st complex root: 2nd [Reason: The first complex root is $-0.2 + 0.4i$ as per the given example, hence it lies in the 2nd quadrant]

Example2:

(Please take the input and display the output as shown)

Input:

a:1

b:2

c:-3

Output:

Equation: $x \hat{} 2 + 2x - 3 = 0$

Roots: Real **[20]**

2. Steel is an alloy that is built of iron with typically a few tenths of a percent of carbon for improving its strength as well as fracture resistance in comparison to other forms of iron. Let us consider that a certain grade of steel is graded as per the given condition:

   Firstly, the hardness of the steel should not be less than equal to 60. Secondly, the content of the carbon in the steel should be greater than 0 and less than 0.65.

   Finally, strength of tensile must be greater than 6400

   The grades are as follows: Grade is 10 - on satisfying all the three conditions

   Grade is 9 - on satisfying conditions (ii) and (iii)

   Grade is 8 - on satisfying conditions (i) and (ii)

   Grade is 7 - on satisfying conditions (i) and (iii) are satisfied

   Grade is 6 - if none of the conditions are satisfied

   Grade is 5 - if only one condition is satisfied

   Write a program, to take the values of hardness, carbon content and tensile strength of the steel under consideration as input from the user and output the grade of the steel. You can check for the respective grade of the steel in the same order as given in the question.

   Example1:
   (Please take the input and display the output as shown)

   Input:

   Hardness: 80

   Carbon content: 0.30

   Tensile strength: 5000

   Output:

   Grade: 8 **[20]**

3. An aeroplane can travel at a speed of 700km/h from a source A either eastwards or westwards. If it travels east, the time advances at a rate of 1 hour per 500 km and if it travels west time recedes back at the equal rate, with respect to the starting point A. Write a program to take input i) the starting time of the aeroplane in hours and minutes only (you can consider 24 hour clock so 1PM becomes 13 hour 0 minutes), ii) the duration of the flight (you can assume the maximum duration to be 6 hours) and iii) a character representing eastward or westward travel (like 'e' for eastwards and 'w' for westwards). Hence perform the following,

(A) Check whether the starting time is a valid time or not. If the starting time is invalid, print "Error" and you can exit from the program.

(B) Depending on the duration of flight and eastward or westward direction of travel, display the local time of arrival (i.e the local time at the arrival point) in the same 24 hour format. Consider the case of adjusting the time of arrival as per the standard 24 hour day format and take care of hour overflow (in case of eastward travel) or underflow (in case of westward travel).

(C) Also print "same day" or "previous day" or "next day" depending on the day of arrival with respect to the starting day

Assume that the aeroplane travels in the given speed uniformly from source to destination. Also you can assume all integer values.

Example1:
(Please take the input and display the output as shown)

Input:

Starting hour: 14

Starting minute: 15

Duration: 5

Direction: e

Output:

Starting time is valid

Arrival time hour: 21

Arrival time minute: 15

Arrival day: same day

Example2:
(Please take the input and display the output as shown)

Input:

Starting hour: 26

Starting minute: 15

Duration: 6

Direction: w

Output:

Starting time is invalid **[30]**

4. The litres of fuel sold on a day depends on which day of the week it is. Consider Monday as Day 1, Tuesday as Day 2, ..., Sunday as Day 7. Let s be the number litres of fuel sold on a day and v be the number of vehicles sighted on that day across the highway. The number of litres can also be a fraction.

(A) For Monday, **s** is given by one-third of the number of vehicles sighted in the highway across, i.e **v**

(B) For Wednesday, **s** is **2v/7**, rounded off to the nearest lower integer.

(C) For Tuesday, Thursday, and Friday, **s** is $\mathbf{v^2 + 2v}$

(D) For the weekends, s is thrice the formula as that used for Tuesday

Take the day (1 to 7) and **v** as input and calculate the amount of fuel sold **s** on that day, and display it. Also calculate the total price of the fuel sold on that day by assuming that the price of fuel = Rs 120/litre and the GST on the fuel as 25% on the base price. **You HAVE to use 'switch-case' for this question**.

Example:

(Please take the input and display the output as shown)

Input:

Day: 3

Vehicles sighted: 100

Output:

Fuel sold '**s**': 28

Total price: 4200

[Reason: 28*120 + 25% GST on (28*120)] **[30]**

Department of Computer Science and Engineering

Assignment 3

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).
- If you do not follow the instructions, your marks may be deducted.

Answer Part A (compulsory) and any two questions from Part B.

[20 + 40 + 40 = 100 Marks]

Part - A
[Compulsory]
[20 marks]

1. You have 'n' number of balls numbered from 1 through n (1,2,3,.....,n). You are playing a game where you select any two cards each of which has some integer number printed over it. Your task is to choose only those balls whose associated numbers are a multiple of either of the two numbers printed on the cards that you have selected (i.e any one or both the card numbers). Write a program for this. Your input should be three integer numbers, 'n' representing the number of balls and 'x' and 'y', each representing one of the two numbers on the cards that you select. Your output should be those balls (i.e the number associated with the balls) that satisfy the game criteria. Print "Game Lost" in case no balls can be selected. You can assume all positive integer values including 1 but excluding 0 for 'n', 'x' and 'y'.

   Example 1:

   Input:

   n = 20

   x = 4

   y = 7

Output:

4

7

12

14

16

20

[Reason: The numbers on the output balls all satisfy the condition that they are either a multiple of 4 or 7 or both.] **[20]**

Part - B
[Attempt any two questions]
[80 marks]

1. Take input 3 numbers from the keyboard. Find out the number of digits of each of the three numbers. Hence find out whether you can construct a triangle whose three sides have length equal to the number of digits of the three numbers. Also if a triangle is possible, then comment on its type based on the length of its side, i.e whether it is a scalene, isosceles or equilateral triangle. You should display i) the number of digits of each of the three numbers that you have taken as input, ii) print 'Yes' if you can form a triangle with sides equal to the number of digits, 'No' otherwise and iii) display 'Equilateral', 'Isosceles' or 'Scalene' based on the nature of triangle, if you can form the triangle, else print 'Triangle not possible'. To include more digits within a given number, you can assume 'long int' type data while taking input. You may further assume that the input numbers will always fall within the range of 'long int'.

   Example 1:

   Input:

   52

   286325

   74523

   Output:

   2

   6

   5

   Yes

   Scalene

   [Reason: The first three outputs show the number of digits of the three input numbers. You can form a triangle with sides 2,6 and 5, which is a scalene triangle.]

   Example 2:

   Input:

1

23

7564

Output:

1

2

4

No

Triangle not possible **[40]**

2. Consider the following series of numbers, **4,8,14,22,32 . . .** . This series follows a specific pattern. Find out the pattern. Hence take a number 'k' as input and print the first k terms of the series according to the rules of the pattern. Also take another integer 'n' as input and print the difference between the (n+1)'th term and the n'th term of the series. **You must write the logic of the pattern as well as the general formula for its n'th term as comments below your code. You can assume that n starts from 1.** **[40]**

3. Consider the infinite summation formula for calculating the value of 'x' as given below. Take two integer numbers 'n' and 'k' as inputs. Find out the value of 'x' by summing the first 'n' terms according to the formula correct to three decimal places. Hence, multiply the value of x with 'k' and consider the floor of the result, i.e the integer value immediately lower than the multiplication value, and check whether the floor value is a prime number or not. You have to display the value of 'x' correct up to three decimal places and print 'Prime' or 'Not prime' based on the primality of floor(x*k). You can calculate the floor value by any way you feel like. Also make sure to take appropriate variable data types. Read the question clearly, you will get all the necessary details about data types.

$$x = \frac{(1\times 2)^3}{((1\times 2)(1+2))^2} + \frac{(2\times 3)^3}{((2\times 3)(2+3))^2} + \frac{(3\times 4)^3}{((3\times 4)(3+4))^2} + \cdots \tag{1}$$

Example 1:

Input:

n=2

k=677

Output:

0.462

Not prime

[Reason: As n=2, the value of 'x' as per the given formula considering only the first two terms correct upto three decimal places is 0.462. Now 0.462 x k=0.462 x 677=312.774. Consider the floor of the answer, i.e 312, which is not a prime number.] **[40]**

4. Write a program to perform the following tasks,

   (A) Take input two positive integer numbers 'n' and 'k'.

   (B) Hence successively keep on taking positive integer numbers as input until you encounter 'k' multiples of 'n', i.e all the various different multiples of 'n' appearing for a total of 'k' times in the input stream.

(C) Find out the maximum and the minimum among those 'k' multiples of 'n' and display them.

You have to use a do-while loop for this question and cannot use arrays. Do not consider 0 or negative numbers as inputs anywhere.

Initial Input:

n=5

k=3

Successive Inputs:

34

23

25

78

100

32

11

85

(STOP HERE, NO MORE INPUT)

Output:

Maximum multiple: 100

Minimum multiple: 25

[Reason: As n=5 and k=3, you would need to stop taking successive integers as input once you encounter any 3 multiples of 5. In the given input stream, the 3 multiples of 5 are 25, 100 and 85, in that order. You stop taking further input once you encounter a total of 3 multiples of 5, i.e 85 in this case. Among the three multiples, 25, 100 and 85, 25 is the minimum and 100 is the maximum] **[40]**

Department of Computer Science and Engineering

Class Test - 1, Section - 16

Subject : Programming and Data Structure (CS19003)

Date: 31st December 2021 Time: 9:30 AM to 12:00 Noon Marks: 100

Instructions for :

- Give the name of the programs files as <Your roll>_<test number>_<question number>.c. For example, 21XX12345_T1_Q1.c for question 1 of test 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<test number>_<question number>_temp.c. For example, 21XX12345_T1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00 Noon. Beyond that, submission will be closed (No extensions will be granted).

---

1. Write a C program to perform the following:

   (a) Take two binary numbers (as long long int) of equal length as input. You have to check if the inputted numbers are binary or not, i.e each digit of both the numbers should be either 1 or 0. Also count the number of bits in both the numbers. Display an error message if any one of the numbers is not a binary number or the two numbers are not of the same bit length, and exit the program.

   (b) If both the numbers are valid binary numbers and they are of the same bit length, then, perform bitwise XOR operation of the two binary numbers from the least significant bit (i.e the rightmost bit) onwards. Display the bit by bit XOR output on the screen as you process each bit of the two numbers from right to left. You **cannot** use any bitwise operators available in the C library.

   NOTE:- The bitwise XOR output that you will display will actually be in the reverse format of the correct bitwise XOR. That is okay, do not worry. Also you do not need to generate any number or such for the bitwise XOR operation, simply print '1' or '0' side by side as per the correct operation. You can assume that the most significant bit of both the input numbers will be 1.

   [Bitwise XOR rules:
   0 XOR 0 = 0
   0 XOR 1 = 1
   1 XOR 0 = 1
   1 XOR 1 = 0] **[30 Marks]**

   Example 1:

Input:
10010
11001

Output:
11010 [Note that this is the reverse of the actual bitwise XOR]

Example 2:

Input:
10010
1101

Output:
Error

Example 3:

Input:
10230
11011

Output:
Error

2. Write a C program to read a sequence of positive integers to detect all possible non-decreasing subsequences and print the total number of such non-decreasing subsequences and the length of the longest non-decreasing subsequence. Within a series of integer numbers, a non-decreasing subsequence is a contiguous sequence of numbers where all are in non-decreasing order. There may be more than one non-decreasing subsequence within the entire sequence, your task is to print the total number of such subsequences present and the length of the longest such subsequence. You should continuously keep on taking inputs for the sequence until the user enters zero or a negative value. Your program must contain only one loop. Both scanning the next integer and processing the scanned integer should be done in that loop. Write no functions other than main(). Do not use any array.

[**Hint:-** In order to get the length of longest subsequence, think of a counter variable and how it is incremented or decremented based on the new input and last input] **[30 Marks]**

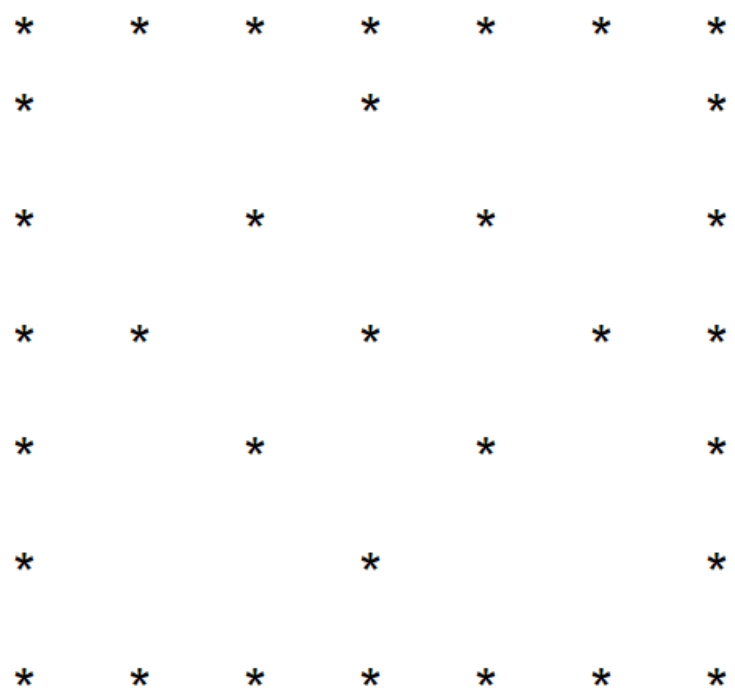
Example:

Suppose you take input the following numbers one by one:-
2, 4, 3, 5, 6, 1, 3, 4, 8, 9, 6, 7, 5, 3, 9, 0 [You stop taking input here]
So the entire valid sequence is: {2, 4, 3, 5, 6, 1, 3, 4, 8, 9, 6, 7, 5, 3, 9}
List of non-decreasing subsequences: {2, 4}, {3, 5, 6}, {1, 3, 4, 8, 9}, {6,7}, {5},{3, 9}
Longest non-decreasing subsequence: {1, 3, 4, 8, 9}

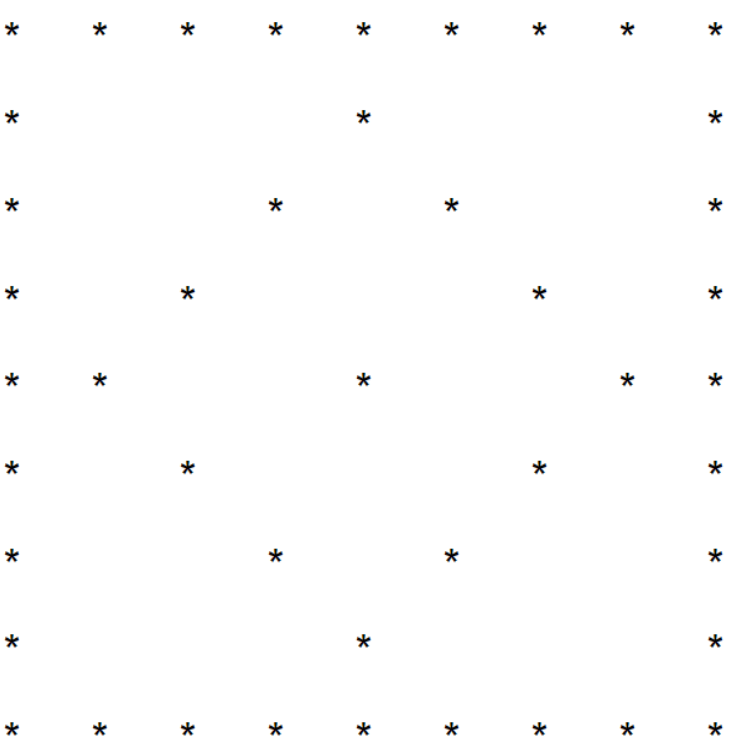
So, your output should be,
Total number of non-decreasing subsequences: 6
Length of the longest non-decreasing subsequence: 5

3. Write a C program to take input a number 'n' and print the following pattern as shown,

For example,for n=3



For n=4,



[40 Marks]

Department of Computer Science and Engineering

Assignment 4

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).
- If you do not follow the instructions, your marks may be deducted.

Answer all the questions. **[20 + 20 + 30 + 30 = 100 Marks]**

1. Write a program to input n integers from the keyboard and store them within an array. Declare another array of size n and copy the contents of the first array into the second array as per the following rules:

   (a) For the elements in the even positions of the first array, into the even positions of the second array in the reverse order. Example: the element in the 1st even position of the 1st array should be placed in the last even position of the 2nd array. Similarly the element in the last even position of the 1st array should be placed in the 1st even position of the 2nd array. For more details see the example given below.

   (b) For all elements in odd positions of the first array, copy the values in the corresponding odd positions of the second array after multiplying that value with the rounded off value of the square root of the sum of squares of all elements to the right of that element in the first array. Round off the square root value to a perfect integer (lower integer in case of decimal part being less than 0.5 and next higher integer in case the decimal part is greater than or equal to 0.5). In case no numbers exist to the right hand side for an odd position number, copy the number as it is in the corresponding odd position of the second array.

   You can input the original array as you wish. Read the question very carefully along with the example shown and understand the logic. For output display the second array only.

   Example:
   (Please take the input and display the output as shown)

   Input:

2 6 7 4 9 3

Output:

28 3 70 4 27 6

[Reason: The even position numbers are copied in reverse order in even positions only. For the odd position numbers, say the number at position 1 i.e '2' of the input array, sum of squares of all elements to the right of '2' in the original array = 191. Square root of 191 is 13.82, so the rounded value is 14, now multiply the original number '2' with 14 to get 28, which is stored in the 1st position of the second array. Apply the same rule for all odd position elements. In case there are no elements to the right of the input array, then copy the value as it is in the second array.]

2. Write a C program to calculate the relative grades of 10 students. The steps to calculate the grade are given as follows:

   (a) Take the marks of the 10 students in an array marks[]. The maximum and minimum marks are 100 and 0 respectively. If the entered marks are outside the range, then error should be displayed, and the system will wait till you enter the marks within the range.

   (b) Calculate the average (μ) of the marks

   (c) Calculate the standard (σ) deviation of the marks, as per the following formula

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}$$

   (d) Hence calculate the relative grades of the marks using the category: 10 if the mark is above (μ + 1.5*σ)
   9 if the mark is above (μ + 1.0*σ)
   8 if the mark is above (μ + 0.5*σ)
   7 if the mark is above (μ)
   6 if the mark is above (μ - 0.5*σ)
   5 if the mark is above (μ - 1.0*σ)
   4 if the mark is above (μ - 1.5*σ)
   Your input should be the marks of 10 students within an array and your output should be the grades of each of the students based on the marks that you have inputted. You may print the grade of each student in a new line and additionally make the output more readable, for example,
   Grade of Student 1 = 9
   Grade of Student 2 = 10
   Grade of Student 3 = 7 and likewise.....

3. Write a C program to perform the following game. You have N number of buckets and each bucket contains some specific number of balls. Take input the value of 'N' from the keyboard and the number of balls in each of the N buckets and store it in an array ball[] of size N. Perform successive iterations whereby at each iteration, you have to find the number of bucket(s) with

the least number of balls and also the least value (say the value be x), and remove 'x' balls from each bucket. You have to continue the game till no balls in any bucket are left. Display the number of bucket(s) with the least number of balls as well as the number of balls remaining in each bucket after you have extracted the least value, i.e x from each bucket. For displaying the number of balls remaining after each iteration, you do not need to resize the ball[] array or declare any other array, simply print the non-zero values within the array side by side after each iteration. You can use only one array to perform all the operations. Consider the example below.

Example:

(Please take the input and display the output as shown)

Input:

Enter Number of Buckets: 5
Enter Positive Number of Balls for Each 5 Buckets: 7 5 2 7 2 [this will be stored in the ball[] array]

Output:

(You would need to print the iteration number as well as shown)

———Iteration-1———

Number of buckets with least number of balls = 2

Numbers of balls in the Remaining buckets = 5 3 5

[Reason: Buckets 3 and 5 have the least number of balls, i.e 2 balls which is the value of x. Now remove x=2 balls from each bucket which provides the balls remaining in the other buckets. Since bucket 3 and 5 have 0 balls remaining you do not print it]

———Iteration-2———

Number of buckets with least number of balls = 1

Numbers of balls in the Remaining buckets = 2 2

[Reason: Continue from iteration 1. Bucket 2 has the least number of balls, i.e 3 balls which is the value of x. Now remove x=3 balls from each bucket which provides the balls remaining in the other buckets. Since bucket 2 have 0 balls remaining and buckets 3 and 5 already had 0 balls from the previous iteration, you do not print it]

———Iteration-3———

Number of buckets with least number of balls = 2

Numbers of balls in the Remaining buckets = NIL [END]

4. You might have observed queues in railway reservation counters. In a queue when the first person leaves the queue, the second person becomes the first member of the new queue. Write a C program to input n characters into a character array of size n. Consider the original queue to be this character array of size n. In each iteration, compute the number of vowels that are present within the current iteration's queue (say this number be 'k') and print all the elements of the queue by skipping the first k elements of the queue. This queue that you have printed becomes your new queue for the next iteration and continue this process until there are no more elements in the queue or there are no vowels remaining in the current queue being considered. For printing the new queue you do not need to perform any deletion or array resizing operation. Rather always keep track of the beginning index of the latest queue for each iteration and advance the index accordingly at the end of iteration. You may use only one array to implement the queue.

   Example:

   (Please take the input and display the output as shown)

Input:

Original queue: a, y, u, g, h, i, e, k, l

[NOTE: While taking input the character array one by one use scanf("<space> %c",&ar[i]) within the loop, where i is the index and ar is the character array. GIVE ONE SPACE BETWEEN THE opening quote and the '%c' within the scanf()]

Output:

(You would need to print the iteration number as well as shown)

——-Iteration-1———

New queue: h, i, e, k, l

[Reason: In the original queue there are 4 vowels, a, u, i and e. Hence k=4. Thus display the queue from after the first 4 elements of the original queue and the displayed queue becomes your new queue for the next iteration]

——-Iteration-2———

New queue: e, k, l

[Reason: Same as above, now considering the new queue from the previous iteration]

——-Iteration-3———

New queue: k, l

[END here, no more vowels]

Department of Computer Science and Engineering

Assignment 5

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).
- If you do not follow the instructions, your marks may be deducted.

[Total Marks = 100 (30 + 30 + 40)]

1. Write a C program to check if the two numbers form a Zazzy pair or not. Two numbers are said to be Zazzy pair, if they satisfy the following criteria as given below:

   (a) Both the numbers should have the same number of digits and their length (number of digits) is even.

   (b) The number formed from the digits of the **first half of the first number (MSB side)** is equal to the reverse of the number formed from the second half of the digits of the second number (LSB side).

   You must use two **functions** for solving this question,

   (a) int checkDigit(int,int) - to check (i) whether two numbers have the same number of digits or not and (ii) Whether their length is even or not

   (b) int isZazzy(int,int) - to check whether the two numbers form a zazzy pair or not. **[30]**

   Example 1:

   Input:

   Number 1 = 123456

   Number 2 = 675321

   Output:

   Zazzy pair

[Reason: The number formed from the digits of the first half of the first number, i.e '123' is equal to the reverse of the number formed from the digits of the last half of the second number, i.e '321']

Example 2:

Input:

Number 1= 123456

Number 2= 321654

Output:

Not a Zazzy pair

Example 3:

Input:

Number 1 = 12345

Number 2 = 54321

Output:

Not a Zazzy pair

Hint:

To provide you with an additional help as to how to frame and formulate your code using functions for this question, consider the code snippet below. Make careful observations as to how you call the appropriate functions at the appropriate places and handle the return values, if any.

```
1   #include<stdio.h>
2
3   int checkDigit(int,int); //Function declaration
4   int isZazzy(int,int); //Function declaration
5
6   int main()
7   { //First, input the two numbers, say the input are stored in variables n and m respectively.
8    //Next proceed to compute for Zazzy pair if and only if the two numbers satisfy conditions a and b of the question.
9
10  int flag = 0;
11
12  if(checkDigit(n,m) == 1)
13  { //Now check whether the two numbers are Zazzy or not
14      if(isZazzy(n,m) == 1)
15      {
16          flag = 1;
17      }
18  else
19  {
20      flag = 0;
21  }
22  }
23
24  else
25  {
26      flag = 0;
27  } //Now if flag == 1, then numbers are Zazzy, so print accordingly, else print not Zazzy.
28  } //end of main()
29
30  int checkDigit(int n, int m)
31  { //Returns 1 if two numbers passed as parameters are of equal length and even number of digits, 0 otherwise
32  }
33
34  int isZazzy(int n, int m)
35  { //Returns 1 if n and m passed as parameters form a Zazzy pair, 0 otherwise
36  }
```

2. Consider a standard deck of cards containing 52 playing cards divided into four sets of spades, heart, diamond and clubs, with each set containing 13 cards, nine number cards with the numbers '2' to '10 printed on them and four letter cards with the letters 'A (Ace)', 'J (ack)', 'K (King)' and 'Q (Queen)' printed on them. Now for simplicity, consider the number card '10' removed from each set, so each set now contains 12 cards in total. Three siblings Alice, Bob, Sarah play a game, where five cards are randomly picked from the deck of cards and given to one of them. In the same manner, five cards are given to each of the remaining two players. The player with the highest sum of the cards wins the game. The cards J (Jack), K (King), Q (Queen), A (Ace) have values of 11, 12, 13 and 1 respectively. For winning the game, only the sum of the card values matter, their corresponding set does not matter. Write a C program following the below points to play this game.

   (a) Write a C function **int counting_cards(char arr[ ])** that computes the sum of the cards of a player as per the character array passed as argument.

   (b) Take three character arrays (char Alice[5], char Bob[5], and char Sarah[5]) to take the cards distributed to each sibling and find the winner. Your input should be either from '2' to '9' (as characters, not int) for denoting the number cards (note that card number '10' has been removed) or 'A', 'J', 'K' and 'Q' for the four letter cards, for each of the three siblings. You must call int counting_cards(char arr[]) from main() for each of the three siblings, determine the winner and display the winner in main() itself. In case there is a tie among sum values **for any two** or all of the siblings, simply print 'game tied'. **[30]**

   Example 1:

   Input:

   Alice: AJ23K

   Bob: 234Q8

   Sharah: 12678

   Output:

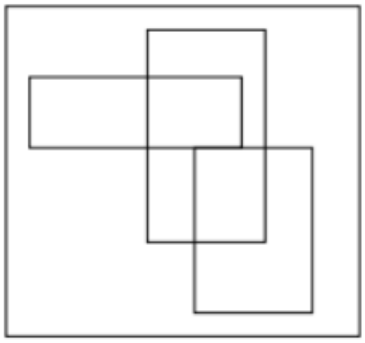
   Alice total: 29

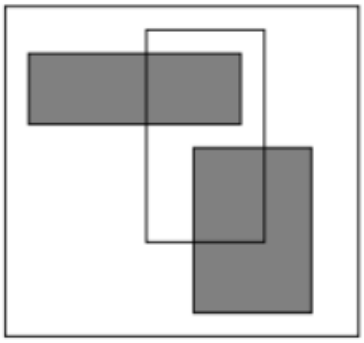
   Bob total: 30

   Sarah total: 24

   Bob wins!

3. Two rectangles are said to overlap if there exists a common point lying inside or on the boundary of both rectangles. Consider the image below, for the first figure all the rectangles overlap, while for the second figure the two shaded rectangles do not overlap. Assume that all rectangles have edges parallel to the x and y axes, and all rectangles lie completely within the first quadrant itself (i.e their corner point coordinates are all positive numbers). The rectangle can be represented by a pair of diagonal points (x1, y1) and (x2, y2), using an array rect[4] = {x1, y1, x2, y2}.





(a) Write a C function **int overlap(float [ ], float [ ])** that receives the diagonally opposite corner points of two rectangles as arguments. The function returns 1 if the rectangles overlap, otherwise 0.

(b) Write a main() program that reads the values of coordinates of diagonally opposite corner points for three rectangles r1, r2 and r3 . For coordinates, consider floating point numbers as well. Hence, for the three rectangles, consider each unique pair of rectangles and print whether the pair overlaps or not. Call the function int overlap(float [ ], float [ ]) to check for each pair. For output, simply print the rectangle pair and its overlap status. **[40]**

For example:

(r1, r2) — Overlaps

(r1, r3) — Does not overlap

(r2, r3) — Overlaps

If all the 3 pairs of rectangles overlap, then you may print (r1, r2, r3) - overalp

---

Department of Computer Science and Engineering

Assignment 6

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).
- If you do not follow the instructions, your marks may be deducted.

[Total Marks = 100 (30 + 30 + 40)]

1. Consider the following recursive formula for calculating the n'th term of a series,

$$\mathbf{a}_n = 2 * (a_{n-1}) + 3 * (a_{n-2}), \qquad a_0 = 1, a_1 = 1$$

The first few terms of the sequence (starting from a0) is 1, 1, 5, 13, 41, 121.....Write a C program to input a number 'n' less than 20 and print the value of an . You must call a recursive function to calculate the value of an. Also in main(), you must keep a check that the input value of 'n' must be less than 20. If the input is greater than 20, then the program must display error 'Out of Range' and wait for the user to enter a valid input again. **[30]**

2. Take an integer array as input from the user. Write a recursive function to find the absolute difference of the consecutive two elements in that array from beginning to end of the array. To solve this problem, you need to write 2 functions: Function-1 computes the absolute difference between 2 numbers and returns the same. Function-2 is a recursive function which computes the successive differences between the elements of an array from beginning to end of an array. Function-2 uses the Function-1 for calculating the difference between 2 numbers, and it should take only the integer array and its size as input arguments.

Example:

Enter the number of elements of the array: 6

Enter a number: 12

Enter a number: 23

Enter a number: 34

Enter a number: 45

Enter a number: 56

Enter a number: 67

The absolute difference of 12 and 23 is: 11

The absolute difference of 23 and 34 is: 11

The absolute difference of 34 and 45 is: 11

The absolute difference of 45 and 56 is: 11

The absolute difference of 56 and 67 is: 11 **[30]**

3. Write a C program using recursive functions to compute the binary equivalent for a given decimal fractional number.

[**Hint:** In the first step, separate integer and decimal parts of a given decimal fractional number. For each part write a separate C recursive function. void int_to_bin(int, int[ ]), void frac_to_bin(int, int[ ]). Consider the max size of the array is 16 for storing the binary equivalents of integer and fractional parts. While printing the binary equivalent of a given decimal fractional number in the main(), only required portions of binary digits to be displayed.]

Example:

Enter a decimal number

13.625

integer part 13

decimal part 0.625000

Binary equivalent:

1101.101 **[40]**

Department of Computer Science and Engineering

Assignment 7

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).
- If you do not follow the instructions, your marks may be deducted.

[Total Marks = 100 (30 + 35 + 35)]

Answer all questions

1. A country has 'n' number of zoos, where each zoo has a certain number of tigers, lions and elephants. It is required to perform some analysis on the number of animals within each zoo of the country. Declare a structure named 'zoo' that contains three integer type values, each to hold the number of tigers, lions and elephants within that zoo. Also declare an array 'zoo_count[ ]' of size n, which is an array of structure of the type 'zoo' that holds the information about all the n zoos of the country. Take, 'n' as input from the keyboard and fill out the array 'zoo_count[ ]' taking the number of tigers, lions and elephants of each zoo as user input. Note that zoo_count[0] holds the number of animals (tigers, lions and elephants) of the first zoo, zoo_count[1] holds the number of animals of the second zoo and so on. Also as zoo_count[ ] is an array of structures, each element of the array is actually a structure of type 'zoo'. After you have taken input, perform the following three operations and print the result.
   - Print the total number of tigers in all the zoos within the county.
   - Print all the zoo numbers (i.e., the index of the zoo_count[ ] array) that contains less than or equal to 6 lions.
   - Print the zoo numbers (i.e., the indices of the zoo_count[ ] array) according to the increasing order of the number of elephants contained within each zoo.

   Example:

   Input:

   Enter 'n' : 3

Enter for zoo 1

Tigers - 4

Lions - 3

Elephants - 10

Enter for zoo 2

Tigers - 2

Lions - 8

Elephants - 5

Enter for zoo 3

Tigers - 8

Lions - 5

Elephants - 4

Output:

Total number of tigers = 14

Zoos with lions less than or equal to 6 = 1, 3

Zoo indices as per increasing order of elephants = 3, 2, 1 **[30]**

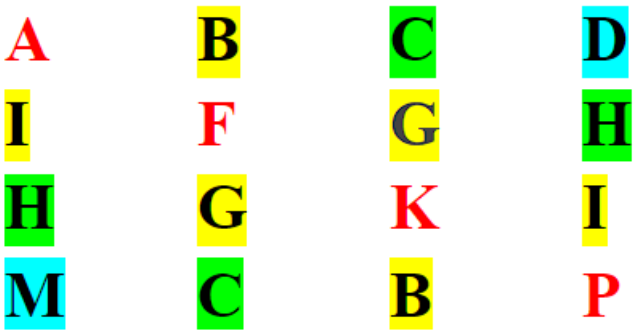
2. A square 2D character array of size nXn is called 'character-mirrored' if it satisfies the following condition.

   - Consider all the left diagonals of the array except the principal left diagonal, the upper right corner element of the array and the lower left corner element of the array. A **symmetric pair** of left diagonals is such a pair which consists of an upper left diagonal and a lower left diagonal, both separated equally from the principal left diagonal. If, for **ALL** symmetric pairs of left diagonals of the array, the characters (elements) of upper left diagonal of each symmetric pair is equal to the reverse order of the characters (elements) of its corresponding lower left diagonal of that symmetric pair, then the array is called 'character-mirrored', otherwise it is not 'character-mirrored'.

For example, consider the 4x4 character array below:

A	**B**	**C**	**D**
I	**F**	**G**	**H**
H	**G**	**K**	**I**
M	**C**	**B**	**P**

Now consider the same array with different color codings to help you understand



Principal left diagonal = [A, F, K, P] (shown in red)

Upper right corner element = D (highlighted in blue)

Lower left corner element = M (highlighted also in blue)

1^{st} symmetric pair (highlighted in yellow both above and below the principal left diagonal):

Upper left diagonal = [B, G, L]

Lower left diagonal = [L, G, B]

2^{nd} symmetric pair (highlighted in green both above and below the principal left diagonal):

Upper left diagonal = [C, H]

Lower left diagonal = [H, C]

Now consider the two symmetric pairs. For the 1st pair the characters in the upper left diagonal are exactly the reverse of the characters of the lower left diagonal. For the 2^{nd} symmetric pair also, the same observation can be made. Hence this particular 2D character array is 'character-mirrored'.

Write a C program to input an n X n 2D character array and print all pairs of symmetric left diagonals. Also, check whether the 2D array is 'character-mirrored' or not. **[35]**

3. Consider a date calculator that performs the following operations as per the following C functions -

   i leap_year(date) - checks if the date belongs to a leap year or not, returns 1 if date is in leap year, returns 0 otherwise.

   ii day_identify (date) - identify the day of the date i.e., Monday, Tuesday. . . It does not return anything, prints the day within the function.

   iii _diff(date1, date2) - calculates the difference between two dates passed as arguments in the number of days and returns it.

   iv date_identify(date1, n)- identify the date that is after n days from the given date. This returns the final date.

   Write a program that performs all the above operations as per the menu selected by the user (see example below). For storing the date, **use a structure** named date that contains three members of integer type: namely day, month and year. You must declare a function for each of the above mentioned operations. For taking input the dates, you can follow the example shown below. No need to keep a separate logic that checks whether the input date is valid or not, you

may assume that the user always enters a valid date.

Example 1:

Input:

– – – – – – – –MENU– – – – – – – –
1. Check leap year
2. Identify corresponding day of a given date
3. Compute difference between two dates
4. Calculate the next date after n days from a given date

Enter your option:
1

Enter date:
Day -12
Month -5
Year - 2022

Output:
The given date is does not belongs to a leap year

Example 2:

Input:

– – – – – – – –MENU– – – – – – – –
1. Check leap year
2. Identify corresponding day of a given date
3. Compute difference between two dates
4. Calculate the next date after n days from a given date

Enter your option:
2

Enter date:
Day - 12
Month - 1
Year - 2022

Output:

The corresponding day of the given date is Wednesday

Example 3:

Input:

– – – – – – – –MENU– – – – – – – –

1. Check leap year
2. Identify corresponding day of a given date
3. Compute difference between two dates
4. Calculate the next date after n days from a given date

Enter your option:

3

Enter first date:

Day - 12

Month - 2

Year - 2022

Enter second date:

Day - 29

Month - 1

Year - 2022

[NOTE:- The first date need not necessarily always be before the second date as per calendar order. Please consider this fact]

Output:

There are 14 days between the given two dates.

Example 3:

Input:

– – – – – – – –MENU– – – – – – – –

1. Check leap year
2. Identify corresponding day of a given date
3. Compute difference between two dates
4. Calculate the next date after n days from a given date

Enter your option:

3

Enter first date:

Day - 12

Month - 2

Year - 2022

Enter second date:

Day - 29

Month - 1

Year - 2022

[NOTE:- The first date need not necessarily always be before the second date as per calendar order. Please consider this fact]

Output:

There are 14 days between the given two dates.

Example 4:

Input:

– – – – – – – –MENU– – – – – – – –

1. Check leap year
2. Identify corresponding day of a given date
3. Compute difference between two dates
4. Calculate the next date after n days from a given date

Enter your option:

4

Enter date:

Day - 28

Month -1

Year - 2022

Enter number of days: 12

Output:

The corresponding date after 12 days is 9/2/2022 [Print in format dd/mm/yyyy] **[35]**

Department of Computer Science and Engineering

Class Test - 2, Section - 16

Subject : Programming and Data Structure (CS19003)

Date: 4^{th} February 2022 Time: 9:30 AM to 12:00 Noon Marks: 100

Instructions for :

- Give the name of the programs files as <Your roll>_<test number>_<question number>.c. For example, 21XX12345_T1_Q1.c for question 1 of test 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<test number>_<question number>_temp.c. For example, 21XX12345_T1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00 Noon. Beyond that, submission will be closed (No extensions will be granted).

---

1. Write a C program that stores n integer values in a 1D array from the user and search for a key value in the array. If the key value is present in the array, shift the elements present on the right of the key by one position towards the left and finally store the key value at the end of the array. Otherwise, display "The key element not found!". The key should also be taken as input.

   Example 1:

   Input:
   Enter the value of n: 5

   Enter the values in the array:

   1

   6

   2

   7

   3

   Enter the key value: 6

   Output:
   1 2 7 3 6 **[15 Marks]**

2. A set can be represented programmatically by a 1D array of elements, where no repetition is permitted. Now let us consider that there are n different sets, each having a maximum cardinality of m elements. This can be represented by an nXm 2D character array, where each row of the array represents one of n sets and each set (i.e each row) has a maximum size of m. Declare

such a 2D character array taking the values of 'n' and 'm' from the keyboard. Hence fill up the array by filling up each set (i.e each row) with the respective set elements. You may consider that the valid set elements contain only printable ASCII characters, i.e A-Z, a-z, 0-9 and special characters. Note that two sets (i.e any two rows of the 2D array) may not be of the same size. Each set can contain any number of elements less than or equal to 'm' (the maximum cardinality of each set) **including zero elements** (the null set). The **space character** (ASCII value = 32) is used to denote an empty cell of the 2D array which contains no element. Thus, for each set (i.e each row), if the set contains less than 'm' elements, then all the cells in that row are filled with **space** after the end of that set (i.e after the last valid element of that set). Write the following C functions to perform the following operations on any two sets, where the 2D array A[][] denotes the collection of all the sets. These functions are **not recursive**.

- void Union (char A[ ][m], int i, int j): To display the union of two sets A[i] and A[j].
- void Intersection (char A[ ][m], int i, int j): To display the intersection of two sets A[i] and A[j].
- void Difference (char A[ ][m], int i, int j): To display the difference of two sets A[i] and A[j] (A[i] - A[j]).

In the main() function, the user should insert the values of n and m and fill up the 2D array with set elements. Following this, the user should insert the value of i and j representing the ith and jth set and call the respective functions as mentioned above to perform the set operations and display the output within that function itself. In case the output of the set operation results in a null set, print "**Null set**" as output for that particular function call.

Example:

The following 5 x 6 array represents 5 sets where each set can store at most 6 elements.

A	**B**	**@**	**2**	**i**	**<space>**
v	**n**	**7**	**8**	**<space>**	**<space>**
S	**&**	**5**	**B**	**2**	**<space>**
1	**2**	**3**	**5**	**G**	**H**

Now for example i = 1 (the 1st set) and j = 3 (the 3rd set), thus the output of the set operations will be,

Union: **[A, B, @, 2, i, $, &, 5]**

Intersection: **[B, 2]**

Difference: **[A, @, i]** **[25 Marks]**

3. Write a menu-driven C program that declares two structures named :

   a. Circle - having three integer members, radius r, x, and y, where (x,y) represents the center.

   b. Triangle - having *perpendicular*, *hypotenuse*, and *base* as the members of type integer

   Perform the following operations according to the functions as defined below.

   1. check_circle(struct Circle c1, struct Circle c2)- Check if the two circles intersect or touch each other or not. If they intersect then return 2. If both the circles touch each then return 1, otherwise 0.

      [**Formula**: Firstly, we calculate the distance between centers C1 and C2 as

      dist = $sqrt((x1 - x2)^2$ + (y1 - y2)^2).

R1 and R2 are the radii of the circles.

Next, we check the following three conditions.

1. If dist == R1 + R2

   Circle A and B are touching each other.

2. If $dist > R1 + R2$

   Circle A and B are not touching each other.

3. If $dist < R1 + R2$

   Circles are intersecting each other.]

2. check_triangle(struct Circle c1, struct Triangle t1) - Firstly, check if the triangle is a right angle triangle or not. If it is not a right angle triangle, then returns 2. Otherwise, check if the area of the circle is equal to that of the incircle of the triangle or not after rounding off their respective values. If the area of the circle is equal to the area of the incircle, then returns 1, otherwise 0.

   [**Formula**: the area of the incircle will be $\pi * ((P + B–H)/2)^2$, where P, B, and H represent perpendicular, base, and hypotenuse respectively.]

   [**Display**: If 2 is returned, display "Not a right angle triangle" If 1 is returned, display "The area of the circle is equal to the area of the incircle of the triangle" If 0 is returned, display "The area of the circle is not equal to the area of the incircle of the triangle"]

   Your program must keep on running continuously until the user inputs 'n'. After each computation, prompt the user to continue again or not.

Example:

Input:

– – – – – – – – – –MENU– – – – – – – – – –

1. Check circles
2. Check triangle

Enter your choice: 1

Enter radius, x, and y coordinate values for the circle A: 4 2 3

Enter radius, x, and y coordinate values for the circle B: 7 15 19

Output:

Circle A and B are not touching each other.

Do you want to continue: y

Input:

– – – – – – – – – –MENU– – – – – – – – – –

1. Check circles
2. Check triangle

Enter your choice: 2

Enter the values of p, b, h of the triangle : 3 4 5

Enter the radius value of the circle: 3

Output:

The area of the circle is not equal to the area of the incircle of the triangle.

Do you want to continue: n **[25 Marks]**

4. Write a C program that performs the following operations. It prints 'n' rows of binary bit patterns ('0's and '1's) by following the rules as given below.

   (a) Consider that the first row always starts with value 0.

   (b) In each subsequent row, considering the previous row, replace each occurrence of 0 in the previous row with 01 in this row, and each occurrence of 1 in the previous row with 10 in this row. Continue like this for every subsequent row till you have 'n' rows. Consider the example given below to understand this concept clearly.

   Write a program to input the value 'n' and print the n different rows. You must use a **recursive C function** *int theithSymbol()* to generate each successive row from the previous one. You are free to use any additional data variables and arrays (1D array, 2D array etc) that you may wish. You are also free to choose whether you would print the i'th row within the recursive function, or you would print the i'th row in main() and the recursive function will only return the i'th row output to main() (say for example by filling up an array representing the i'th row). But you **must** ensure that the logic that you use to get each successive row from the previous one is recursive. The order of arguments of the recursive function is also your choice.

   Example:

   Input:

   Enter the value of n: 5

   Output:

   Value of Row 1: 0

   Value of Row 2: 01

   Value of Row 3: 0110

   Value of Row 4: 01101001

   Value of Row 5: 0110100110010110

   [Note how each row is being generated from the previous row by replacing 0 with 01 and 1 with 10. The first row contains 0 by default] **[35 Marks]**

Department of Computer Science and Engineering

Assignment 8

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).
- If you do not follow the instructions, your marks may be deducted.

[Total Marks = 100 (30 + 30 + 40)]

Answer all questions

1. Take input a long sentence that contains an arbitrary number of words. Two words are supposed to be separated by only one blank space character. Also a valid word consists of only the ASCII characters 'A-Z' or 'a-z' and nothing else. However the sentence that you will be taking as input may not follow this rule, i.e there may be more than one blank spaces between two words as well the words themselves may contain invalid characters. Your task is to refine this invalid input to get the correct valid string **in place**, i.e that you CANNOT use any additional array to perform this operation, you have to process within the same input array with perhaps one or two additional character variables. Print the final valid string as well. You can assume that any sequence of characters 'A-Z' or 'a-z' forms a valid word although it may not exist in the dictionary [No need to consider full stop also at the end]. You are **NOT** allowed to use any in-built string functions.

   Example:

   Input:

   Acceler56ation is def9ine%d as r34@at56e 78of chang4e of ve2@locity

   Output:

   Acceleration is defined as rate of change of velocity **[30]**

2. Camel case notation is a special type of typography where multiple different words are combined together as one word without any space in between AND the first letter of each different word is uppercase while the remaining letters are lowercase. For example 'MySchool' is a camel case notation consisting of two separate words 'my' and 'school', while 'IAmHappy' is a camel case notation containing three words, 'I', 'am' and 'happy'. Now consider the following two functions to perform the tasks as mentioned.

   - **int numberOfWords(char *camel)** - takes input a camel case string char *camel as argument and returns the number of constituent words.
   - **void camelToReverse(char *camel, char *reverse)** - Takes input two arguments, one camel case string (char *camel) and another empty string (char *reverse). Hence it fills up the empty string with the constituent words of the camel case string **in the reverse order** (as it appears in the camel case string) with a space in between the constituent words and terminated with null. For example if the camel case string is 'IAmHappy', then reverse[ ] will contain 'Happy Am I', which is also a null terminated string.

   Write a C program to input a camel case string containing any number of constituent words. First print the number of constituent words of the input string by calling the appropriate function within main(). Then call the function camelToReverse() to generate the reverse string of the constituent words and print them within that function only. You can assume that the input is always a valid camel case string. Note that you can declare character arrays inside main() for taking the string input and initializing the empty string, BUT you must work with pointers to string within the two functions, i.e while calling the functions you must pass the base address of the character arrays and work with pointers inside the respective functions. You are **NOT** allowed to use any in-built string functions except probably strlen() **ONLY**.

   Example:

   Input:

   'HowDoYouDo'

   Output:

   Number of words: 4

   Reverse string: 'Do You Do How' [using pointers]

   [NOTE the space in between the constituent words] **[30]**

3. It is unsafe to send data directly from one place to another. Therefore, we use encryption to convert the files/strings into an unreadable format which needs some key/ method to decrypt to get the actual file/string. Write a C program to encrypt a text string following the given steps:

   1. Take a string as input (the string can contain alphabets and numbers)

   2. Take the 8-bit binary representation of each character in the input string (you can use int $i = (int)c$; if c is a char) and concatenate all these bits to get an intermediate representation of the whole input string.

   3. Then take each groups of 6-bits in the intermediate representation (starting from left-end towards the right) and map it using the following rules:

   $0 => A, 1 => B, ..., 25 => Z$ (Capital Letters)

   $26 => a, 27 => b, ..., 51 => z$ (Small Letters)

$52 => 0, 53 => 1, ..., 61 => 9$ (Numericals)

Example:

Input:

IIT Kharagpur

[Intermediate steps:

1. (int) 'I' is 73
2. 8-bit binary representation of 73 is 01001001
3. Hence, intermediate representation is 0100100101001001...
4. After converting first 6-bits from the intermediate representation, we get $(010010)2 => (18)10 => S$]

Output:

SUlUIEtoYXJhZ3B1cC

Input:
C Programming

Output:

QyBQcm9ncmFtbWluZC

[Hint: You can consider a temporary integer array of size (8*number of characters of string) where you can store the intermediate bit representation, each bit as an integer 1 or integer 0 stored in each cell of the array] **[40]**

Department of Computer Science and Engineering

Assignment 9

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).
- If you do not follow the instructions, your marks may be deducted.

[Total Marks = 100 (30 + 30 + 40)]

Answer all questions

1. A dynamic array is an array whose size is only known during the runtime of a program. You can use the following two functions, malloc() or calloc() to define dynamic arrays. **A strict sawtooth array** is an array where the values of the sequence of elements should follow the up-down principle. The successive elements of the array should not be the same. The successive differences across the array should follow alternate +ve and -ve values. For example (1,4,2,5,4,8,6,10,9) and (-10,-15,0,-2,6,3) are strict sawtooth arrays of size 9 and 6, respectively, but (1,5,7,3,6) and (34,23,23,44) are not strict sawtooth arrays. In this program you have to use the concept of dynamic arrays to perform the following tasks with respect to sawtooth arrays.
   - Prompt the user to enter the size of the array as 'n'. Declare a dynamic integer array of the same size.
   - Take input 'n' different integers into the array from the keyboard.
   - Check if the array is a sawtooth array or not. Use the function **int isSawtooth(int *)** that takes as argument a pointer to the array and returns 1 if the array is sawtooth, otherwise it returns 0. Print the appropriate message on the console.
   - If the array is NOT a sawtooth array, then find out the first position in the array where the sawtooth condition is violated. Use the function **void position(int *, int *)** which takes in two pointer arguments, first one is the pointer to the array and the second one a pointer to another integer variable declared in main. The position where the sawtooth condition is violated is written into the variable pointed by the second argument (Note that this function should NOT return the position). Also this function should be called only if the array is not sawtooth.

- The program then frees the dynamic array created. It asks whether the user wants to continue again or not. It prompts the user to input 'Y' (for Yes) or 'N' (for No) to continue the program. If 'Y' is entered, then the steps are repeated again and if 'N' is entered, the program exits. The array has to be dynamically allocated each time the program runs.

Example:

Enter n: 9

Enter the array:

1,4,2,5,4,8,6,10,9

The array is a strict sawtooth array.

Do you want to continue (Y/N): Y

Enter n: 5

Enter the array: 1,5,7,3,6

The array is not a strict sawtooth array.

First violating position: 3

Do you want to continue (Y/N): N **[30]**

2. A user is asked to input two n-dimensional vectors but the dimension is not known initially. We use a structure to define a vector type variable, which contains two elements, an integer number denoting the dimension of the vector the structure is currently storing and a float pointer that points to a dynamically allocated array that stores the vector components corresponding to each of the n dimensions. Take input a number n and correspondingly take input two n-dimensional vectors, storing the dimension and the components of each vector into two corresponding vector type struct variables. Note that before taking input the vector components, remember to initialize the array within the struct variable which will hold the componants. Write the C-functions to perform the following.

   - **struct Vector VectorSum(struct Vector *v1, struct Vector *v2)**: Performs the sum of the two vectors of the same dimension and returns another structure of type vector that holds the sum. For two n-dimensional vectors with components [x1, x2, x3,...,xn] and [y1, y2, y3,...,yn], their sum is another n-dimensional vector [x1+y1, x2+y2, x3+y3,....,xn+yn].
   - **int OrthogonalTest(struct Vector *v1, struct Vector *v2):** Returns 1 if two vectors v1 and v2 are orthogonal, 0 otherwise. Two n-dimensional vectors with components [x1, x2, x3,...,xn] and [y1, y2, y3,...,yn] are said to be orthogonal if their inner product (dot product) is zero, x1.y1+x2.y2+x3.y3+....+xn.yn = 0.

   After taking input the two vectors, first print both the vectors on console in the format **'(n, [.....])'**, where n represents the dimension and put the dimension components within the square brackets. Hence call the above mentioned functions from main() and print the results returned.

   [30]

3. You can put multiple strings within a 2D character array where each row of the character array holds one string. For example char arr[3][100] can hold three strings, each of length of maximum 100 characters and the first string is identified by arr[0], the second string by arr[1] and so

on. Declare a 2D array **string[5][1000]** that can hold 5 strings of maximum 1000 characters each and take input the 5 strings. Each string is a sentence containing any number of words. Words consist of alphabets (you may consider all lowercase characters for simplicity) only and no numericals or special characters, and two words are separated only by one blank space. No need to use fullstop also to mark the end of the string. Hence perform the following operations as defined by the following functions.

- **void characterCount(char *str, int *vowel, int *consonant)** - It counts the total number of vowels and number of consonants of a string pointed by char *str, and writes the vowel count to the address pointed by int *vowel and similarly the consonant count to the address pointed by int *consonant. A space is neither a vowel nor a consonant.
- **int wordCount(char *str)** - It returns the number of words of the string pointed by char *str.
- **void excessWord(char *str1, char*str2)** - This function first calls wordCount() to get to know the number of words of each of the two strings char *str1 and char *str2. Hence it prints the excess words that are present in the longer string. For example if the first string contains 4 words and the second string contains 6 words, then it prints words 5 and 6 of the second string. If both the strings contain equal number of words, then it prints "No excess word".
- **void str_pair_compute(char **arr, int number_of_strings)** - This function takes as input the base address of the entire 2D array containing all the strings and another integer denoting the number of strings within the 2D array. Hence it displays the pairs of strings which contain the same words. The words need not be in the same order. In case no two strings contain the same words, then it prints "Not Applicable". This function may also use wordCount() if required. Consider the example as shown. Suppose the 5 strings in string[][] are,
  - abcd sdfg qwerty oiuyt
  - cfr ojhg nsert
  - evjs irjcl perry
  - nsert cfr ojhg
  - qwerty sdfg abcd oiuyt

  Here, (string[0] and string[4]) and (string[1] and string[3]) contain the same words. So the function displays (1,5) and (2,4) as output, i.e the 1st string and 5th string and 2nd string and 4th string.

From main() after taking the 5 string as input, first call characterCount() for each of the 5 strings to print the vowel and consonant count within main() itself. Declare integer variables as required. Hence take input two numbers between 1 and 5 that represent any two strings and call excessWord() that prints the excess words among the two strings. Note that excessWord() in turn calls wordCount(). It is upto you how you wish to display the different outputs, but it should maintain the specifications as given. Finally call str_pair_compute() to display the pairs of strings which have the same words in different orders. **[40]**

Department of Computer Science and Engineering

Assignment 10

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).
- If you do not follow the instructions, your marks may be deducted.

[Total Marks = 100 (30 + 30 + 40)]

Answer all questions

1. In a doubly linked list, each node is given two pointers, the first pointing to the next element in the list and the second to the previous element in the list. The next pointer of the last node and the previous pointer of the first node should be NULL. Describe a suitable C data type to store a node in a doubly linked list. Suppose that each node in the list stores an integer key value. Now, consider a **sorted doubly linked list of integers in ascending order**.
   - Write a C function to insert an integer 'x' in 'L' in the sorted order. If x is already present in L, then the insertion produces no change. If x is absent from L, it is inserted in the appropriate position so that after the insertion, L continues to remain sorted.
   - Write a C function to return the **minimum search distance** for a particular element within L. The minimum search distance for an element 'x' within 'L', if 'x' is present in 'L', refers to the minimum of the distances from the front and rear end of the list to the particular element 'x'. For example, if the list contains (2,4,6,8,9,10), then the minimum search distance of element '9' is 2, as distance from front end to '9' = 5 and distance from rear end to '9' = 2, and min(2,5) = 2. If 'x' is not present in 'L', then search distance returns -1. If there are duplicate elements 'x' within 'L', then the first occurrence of 'x' from both the sides should be considered as the respective distance and the minimum of that is returned. If the distance from both the left end and right end is the same (i.e the element is in the middle of an odd length list), then just that distance should be returned. Note that min(a,a) = a only. The search element should be taken as input from the keyboard and passed to the function as one of its arguments. You **HAVE** to use the concept of doubly linked in your logic for this function.

- Write a function to print the sorted linked list starting from the head node. If the list is empty, then print 'List empty'.

  For each of the above mentioned functions, write the function name, function definitions and function arguments as per your choice, but make sure that the functions perform the tasks as specified.and one of the arguments must be the head pointer of the linked list. In main(), write a menu-driven code that provides the user with four choices, insertion, finding minimum search distance, printing the list and exiting the program. Based on the user choice, the appropriate function is called. The menu is run continuously until the user wishes to exit. For insertion and search distance, the respective element is also taken as input. The linked list initially starts empty, with 0 number of nodes. **[30]**

2. Define a linked list where the head node sits in the middle and you can go either to the left or right of the node depending on the user input. The head node contains two pointer variables left and right to traverse either to the left or right of the node. All other nodes contain either the left pointer or the right pointer. Write a program to create such a linked list by inserting new nodes either to the left or to the right of the head node. Logically, the list looks like as shown below,

$$... \leftarrow A \leftarrow B \leftarrow HEAD \rightarrow C \rightarrow D \rightarrow ...$$

Write a C program to perform the following tasks.

- Let the user choose continuously from the keyboard (until '-1' is pressed) the new element to be added and which direction the new insertion should happen.
- Depending on the choice, create a new node containing either the left or right pointer.
- Start from the head and traverse in the appropriate direction till the end of the respective left/right sub chain to insert a new node there.
- At the end display both the left sub-chain and the right sub-chain.

You should write two functions ***insertLeft(node* head, int data)*** and ***insertRight(node* head, int data)*** that inserts a new node either to the left or right of the head respectively. You can assume that the nodes contain only integer data. You can further assume that '1' represents left and '0' represents right as choice during user input.

Example:

Enter data: 10

Enter direction: 1

Enter data: 20

Enter direction: 0

Enter data: 30

Enter direction: 0

Enter data: 40

Enter direction: 1

Enter data: 50

Enter direction: 0

Enter data: -1 [NOTE: '-1' is the exiting condition for taking further inputs]

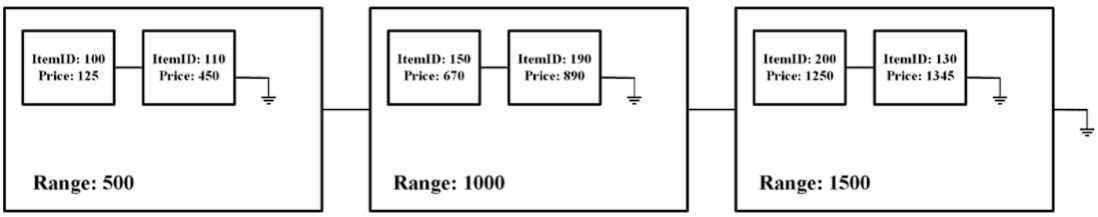
Output:

Left linked list: 10 → 40 → NULL

Right linked list: 20 → 30 → 50 → NULL

[You have to print the arrow heads and the word 'NULL' to clearly represent the list] **[30]**

3. You must have done online shopping where you can filter different items based on a number of parameters, one such being sorted as per the range of price of the item, i.e whether the price lies in the range, say, (0-500), then (501-1000), (1001-1500), (1501-2000) and so on. We can think of each range to be a bucket and within each bucket all the items whose price falls in the respective range are kept. In order to build such a filtering system, we can take the help of nested linked lists. Consider the following diagram below.



As per the diagram shown, there is one outer linked list, and within each node of the outer list there is an inner linked list. Each node of the outer linked list denotes a bucket as per the price range in increasing order, starting from 0. It holds an integer variable 'Range' which denotes the upper limit of the price range for that bucket, the lower limit starting from after the previous bucket's range. Overall the price starts from 0 (lower limit for the first bucket) and each successive bucket has a range of 500 units of currency. That is, if the value of 'Range' is 1000 for a bucket, this means that bucket holds all the items whose prices fall in the range 501 to 1000. The inner linked list denotes the list of all the individual items whose price falls within the range for that particular bucket. Each node of the inner list consists of two integer variables, 'ItemID' denoting the numerical ID of the item and 'Price', denoting the price of that particular item. Within each bucket, the inner list of items is also sorted in increasing order of their prices.

Write a C program to build such a filtering system.

- Define the suitable structure variable that can form such a nested linked list. In particular, the structure variable itself should be a nested structure to incorporate the outer and inner lists.
- Initially, the entire list is empty, i.e no elements are present. Write a function ***void insert(Bucket *head ,int itemID, int Price)***, to insert a new item into the linked list as per the filtering order, where 'Bucket *head' points to the head of the outer linked list, 'itemID' refers to the ID of the particular item and 'Price' refers to the individual price of that item. You must first locate the appropriate bucket node of the outer list where the

item has to be kept, then within that bucket you have to insert that item in the correct position of the inner list. If the appropriate bucket node is not present within the current list, then add a new bucket node that can hold that item, along with any intermediate bucket nodes that may be required. This is required because the buckets have a range of 500 only. The intermediate bucket nodes can hold an empty inner list, if no items for that bucket are present. Keep in mind that both the outer and inner lists are sorted in ascending order, and must remain sorted after each insertion.

- From main(), take input the itemID and price of 'n' such items and insert them one by one within the filtered list by calling insert(). You may assume that for all items, minimum price is 0 and maximum price is 5000.
- Finally after taking input is complete, print the entire list, as per the format below.

Example sample output (refer to the diagram):

Bucket 1:

Lower limit: 0

Upper limit: 500

Items:

(100,125) [NOTE: The item tuple is of the form (itemID,Price)]

(110,450)

Bucket 2:

Lower limit: 501

Upper limit: 1000

Items:

(150,670)

(190,890)

Bucket 3:

Lower limit: 1001

Upper limit: 1500

Items:

(200,1250)

(130,1345) **[40]**

Department of Computer Science and Engineering

Class Test - 3, Section - 16

Subject : Programming and Data Structure (CS19003)

Date: 11^{th} March 2022 Time: 9:00 AM to 11:00 AM Marks: 100

Instructions for :

- Give the name of the programs files as <Your roll>_<test number>_<question number>.c. For example, 21XX12345_T1_Q1.c for question 1 of test 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<test number>_<question number>_temp.c. For example, 21XX12345_T1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00 Noon. Beyond that, submission will be closed (No extensions will be granted).

Part-A [Compulsory]
30 Marks

1. Take input an integer number of 'n' digits and check whether the number is a **Narcissistic number** or not. A Narcissistic number is a number which is equal to the sum of its digits, each raised to the power of the number of digits of the original number. For example, 153 is Narcissistic since it is of 3 digits and $153 = 1^3 + 5^3 + 3^3$. Similarly 1634 is also Narcissistic as $1634 = 1^4 + 6^4 + 3^4 + 4^4$. You must use the following function,

   - **int isNarcissistic(int n)** - checks whether a number is Narcissistic or not. Returns 1 if Narcissistic, 0 otherwise.

   You must take the original number as input from main() and call the respective function and display the appropriate result in main(). You may assume that the input will always be a positive integer of commensurate number of digits.

   Example:

   Input:

   153

   Output:

   Number of digits = odd

   Number is a Narcissistic number

Input:

632

Output:

Number of digits = odd

Number is not a Narcissistic number

Part-B [attempt any one question]

70 Marks

2. You have used passwords in all of your online accounts. A good password is an alphanumeric string where the different constituent characters are randomly placed throughout the length of the string. We can use a structure data type with the following attributes to represent a password in a computer.

   struct password:

   - int Length: Length of the password in number of characters
   - int Uppercase :Number of uppercase characters in the password
   - int Lowercase : Number of lowercase characters in the password
   - int Numbers : Number of numerical characters, 0-9
   - int Special: Number of special ASCII characters
   - int Size : Size in bytes of the password (one character is equal to 8 bits)
   - int Strength: Denotes the strength of the password, defined as the number of characters between the first uppercase and the first next lowercase character of the password, both the characters inclusive. For example a password 'XVBghy7J' has a strength of 4 (number of characters from 'X' till 'g'), 'ghfVbn@' has a strength of 2 (starting from 'V' and ending at 'b'). In case a password has no next lowercase characters after the first uppercase character, then strength is 1 (for example 'ghjtJLI' or 'fb$76bhK') and in case the password has no uppercase character at all, the strength is zero (for example, 'fft67%uj')
   - char Content[50] : The actual content of the password, with a maximum length of 50 characters

   Declare an array of structure of type password to hold 'n' different passwords, 'n' taken as input from the user initially. Hence enter 'n' different alphanumeric passwords, each having arbitrary size and random characters, but maximum size of each password is 50 characters. For each of the 'n' passwords, compute the different password attributes as shown above to fill up the parameters of the structure variables within the array. You may assume that the password consists of only lowercase 'a-z', uppercase 'A-Z', numericals '0-9' and special ASCII characters. There should not be any space within the password. Hence for each unique pair of the passwords among all the 'n' passwords (i.e if there are three passwords, then consider the pairs (1-2), (1-3) and (2-3)), compute the similarity index of the password pair. Similarity index for a password pair is defined as the number of password attributes that are equal for both the passwords, starting from the length upto the strength attribute, except the last attribute (char content[]) and expressed as a percentage. Since there are 7 attributes, if all the 7 attributes match for both the passwords, the similarity index is 100%, if none of the attributes match the index is 0%, if only 2 attributes match the index is (2/7)*100 = 28.57% and likewise. Declare appropriate data types. You can use strlen() **ONLY** and no other in-built string functions. You may however

declare and use user-defined functions as you feel necessary.

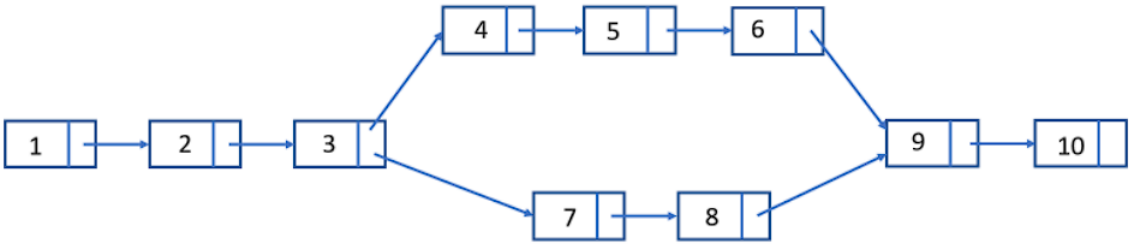
3. READ THE QUESTION CAREFULLY:

   Suppose you want to travel from source location 'a' to destination location 'z'. So you start moving from location $a \rightarrow location\ b \rightarrow \cdots \rightarrow location\ x \rightarrow location\ z$. However, while traveling from location 'a' to 'z', there comes a point where the road diverges and proceeds via two different routes in a branching, which again converges after reaching a particular location. For example, consider the figure given below, location 1 is the source and location 10 is the destination. You can travel from location 1 to location 10 by any of the following two routes,

   $loc\,1 \rightarrow loc\,2 \rightarrow loc\,3 \rightarrow loc\,4 \rightarrow loc\,5 \rightarrow loc\,6 \rightarrow loc\,9 \rightarrow loc\,10$

   loc $1 \rightarrow loc\,2 \rightarrow loc\,3 \rightarrow loc\,7 \rightarrow loc\,8 \rightarrow loc\,9 \rightarrow loc\,10$

   Here, location 3 is the divergence point for the branching and location 9 is the convergence point. Suppose now that you want to know the location of divergence and convergence of the road before traveling so that you can travel through the less congested road. To perform the above operation, firstly define a linked list node named 'Transport' that contains one integer variable 'Loc' to denote the location identity and two pointer variables 'Transport *Next_loc1' and 'Transport *Next_loc2' that point to the next location that one can visit from that node towards the ultimate destination. If from a node there is only one next location that can be visited then that is pointed by *Next_loc1 and *Next_loc2 is NULL. If from a node two locations can be visited next, then the two locations are pointed to by the two pointer variables.



   Write a C program that takes the location information from the user and displays the divergence and convergence location. The user must first input the first route from source to destination and then the second route from source to destination. While taking the input routes, the linked list must be created accordingly. For the first route, create the linked list like a normal single chain linked list. You must use *Next_loc1 as the next pointing pointer and set *Next_loc2 as NULL, as you do not know about the second route, and hence any convergence or divergence yet. Then input the second route between source and destination location by location and first check whether the location is already covered in the first list. If yes, do nothing. If no, then at this point the path diverges, so create a new node for the second branch list and make the *Next_loc2 to point to this node from the diverging node (note that *Next_loc1 of the diverging node already points to the next node of the first branch). Carry on adding new nodes to this second route until again you find a common node with the first list that represents the convergence point. So make the *Next_loc1 pointer to point to the converging node and *Next_loc2 as NULL (reason already given above). Hence for all successive locations of the second route, it is the same as the first one, so do nothing. After taking input and creating the list, print the diverging and converging location. You may assume that the numerical integer 'Loc' is unique for every node and there is only one branching (i.e only one divergence and convergence point) between the source and destination, source and destination always being the same.

Example:

Input:

Enter the first route from source to the destination: 1 2 3 4 5 6 9 10

Enter the second route from source to the destination: 1 2 3 7 8 9 10

Output:

The divergence location is 3

The convergence location is 9

[**Note: 1.** You cannot check the convergence and divergence location at the time of taking the input. While taking the input, create the branched linked list and then check divergence and convergence locations by traversing the list and checking the respective pointers

2. Although you are not printing the entire list as output, **YOU MUST** implement this question with linked lists as defined. You **CANNOT** use arrays or simple scanning through the input sequences to identify the divergence and convergence]
