Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20

Assignment 1 | December 14, 2021

Submission instructions

* Submit one .c file for each part of the assignment. Each of your .c files should be named as:

<your roll number>_A<assignment number>_<part number>.c

For instance, if your roll number is 21CS10023, and if you are presently doing Assignment 1 which has 4 parts, then you should submit 4 separate .c files named as:
21CS10023_A1_1.c   21CS10023_A1_2.c   21CS10023_A1_3.c   21CS10023_A1_4.c

* Submissions must be through the course Moodle, before the end of Lab session (11:55 AM). Late submissions will be penalized / not accepted.

---

1. [10 marks] Write a C program to determine the total amount accrued in a cumulative fixed deposit account for 3 years at the rate of 12 percent per annum if the interest is compounded annually on an initial principal amount of Rs. 1,00,000. The program should print out the total amount.

2. [10 marks] Consider a triangle ABC. Input the <u>lengths</u> of the sides AB, BC, and CA from the user, using scanf statements. Compute and print (i) the perimeter of the triangle ABC, (ii) the square of the area of the triangle. Use Heron's formula to compute the square of the area: https://www.mathopenref.com/heronsformula.html

3. [15 marks] In this problem, you need to compute the derivative of a <u>degree-5 polynomial</u> of one variable x: a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0

Assume that the coefficients (a_i) of the polynomial are integers. Input the 6 coefficients from the user. Print the polynomial, and then compute and print the derivative polynomial. A sample input and output are given below; <u>stick to the format shown below</u>.

```
Enter coefficient of 1: 5

Enter coefficient of x: 2

Enter coefficient of x^2: -2

Enter coefficient of x^3: 7

Enter coefficient of x^4: 0

Enter coefficient of x^5:3

Polynomial: 3*x^5 + 7*x^3 - 2*x^2 + 2*x + 5

Derivative: 15*x^4 + 21*x^2 - 4*x + 2
```

Note that, the numbers given above are only examples. You should write the program such that the coefficients can be any valid integer (different from what is given in the example) and the program still works.

4. [15 marks] In this problem, you need to evaluate trigonometric functions like sin(x) and cos(x), using their respective series expansion formulas. For the series expansion formulas, refer to http://mathworld.wolfram.com/SeriesExpansion.html. Input an angle x (in radians) from the user.

(a) Compute approximate values of sin(x) and cos(x) using the first four terms of their series expansion formulae

(b) Compute sin(2*x) = 2 * sin(x) * cos(x)

(c) Compute sin(2*x) directly using the first four terms of the series expansion formula for *sin*, and compare the value with that obtained in (b).

Your program should take x (in radians) as input, and then print out the values of sin(x), cos(x), and sin(2*x) computed by the two methods described above.

Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20
Assignment 2 | December 21, 2021

Submission instructions

* Submit one .c file for each part of the assignment. Each of your .c files should be named as:

<your roll number>_A<assignment number>_<part number>.c

For instance, if your roll number is 21CS10023, and if you are presently doing Assignment 2 which has 4 parts, then you should submit 4 separate .c files named as:
21CS10023_A2_1.c 21CS10023_A2_2.c 21CS10023_A2_3.c 21CS10023_A2_4.c

* Submissions must be through the course Moodle, before the end of Lab session (11:55 AM). Late submissions will be penalized / not accepted.

---

1. [10 marks] Write a C program that reads two non-negative integers, `a` and `b`. Then it finds which one is larger, and decides whether the larger is divisible by the smaller. It should finally print in the terminal which one divides (or not) which one.

Example 1
```
Enter two positive integers: 18 12
Result: 12 does not divide 18.
```

Example 2
```
Enter two positive integers: 6 18
Result: 6 divides 18.
```

2. [10 marks] Write a C program which reads an integer (say `x`). If the integer is even then print sum of the integers from 1 to that integer (`x`). If integer is odd then print sum of integers from 1 to the twice of that integer (x). For example, if input is 3, then sum would be 21 or if the input is 4, then the sum would be 10.

3. [15 marks] Write a C program that reads the coordinates of the three vertices of a triangle ABC and determines whether the triangle is equilateral or isosceles or neither. In case the triangle is isosceles, your program should also report the two sides that are of equal length. <u>You can use math library functions if required</u>. Check the output of your program on the following triangles:

(1,1), (4,-3), (8,0)
(1,1), (4,3), (8,0)
(1.1,-2.1), (6.6,6.8), (10.0,-7.6)
(2,-2), (-4,4), (-2,0)

---------Example Output---------------
```
Point A: 0,0
Point B: 5,0
Point C: 0,-5
The triangle is isosceles with |AB| = |AC|.

Point A: 2,3
Point B: 3,5
Point C: 5,8
The triangle is neither isosceles nor equilateral.
```

4. [15 marks] In a car manufacturing company, the number of cars manufactured on a day depends on which day of the week it is. Monday means Day 1, Tuesday means Day 2, ..., Sunday means Day 7. Let there be m machines in the company. You have to take m as an input from the user. Let n be the number of cars manufactured on a day.

- For Day 1, n is given by the number of available machines i.e., n is exactly same as m.
- For Day 2, n is 7m/4, rounded off to the nearest integer.
- For Day 3 and Day 6, k machines (k is user input) are kept out for maintenance, and so n becomes 7(m-k)/4, rounded off to the nearest integer.
- For the other days, n is m plus a boosting factor f of m, rounded off to the nearest integer. Assume that f is a positive real number less than 1 and taken as user input.

Given the week day (1 to 7) as input, find the number of manufactured cars (n) on that day. You should use switch-case statements.

---------Example Output----------------

```
Enter the number of machines: 127
Enter the day number (1-7): 2
Number of manufactured cars = 222.

Enter the number of machines: 127
Enter the day number (1-7): 3
Enter no. of machines under maintenance: 25
Number of manufactured cars = 179.

Enter the number of machines: 127
Enter the day number (1-7): 5
Enter boosting factor: .35
Number of manufactured cars = 171.
```

Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20
Assignment 3 | December 28, 2021

Submission instructions

* Submit one .c file for each part of the assignment. Each of your .c files should be named as:

<your roll number>_A<assignment number>_<part number>.c

For instance, if your roll number is 21CS10023, and if you are presently doing Assignment 3 which has 4 parts, then you should submit 4 separate .c files named as:
21CS10023_A3_1.c   21CS10023_A3_2.c   21CS10023_A3_3.c   21CS10023_A3_4.c

* Submissions must be through the course Moodle, before the end of Lab session (11:55 AM). Late submissions will be penalized / not accepted.

---

1. [15 marks] With a positive integer *n* and a real number *x* as inputs, compute the Taylor series for the exponential function e^x, up to *n* terms. For the formula, you can refer to https://www.mathsisfun.com/algebra/taylor-series.html. You should check whether the input *n* is a positive integer. If not, you should print out a suitable error message, and then ask the user to enter the number again. This process should continue until the user enters a valid positive integer. You should try to code in such a way that the number of multiplications is minimized. [Hint: Use a do..while loop for taking the input. To minimize the number of multiplications, think how the (j+1)-th term can be easily computed from the j-th term of the series.]

2. [10 marks] We want to identify all integers between 1 and 1000, whose sum of digits is greater than a given integer *n.* For instance, if *n* = 21, then some integers in [1, 1000] whose sum of digits is greater than *n* are 958, 959, etc. Write a program that takes *n* as input, and then prints out all integers in the range [1, 1000] whose sum of digits is greater than *n.*

3. [10 marks] Write a program that takes an integer *n* between 1 and 9 as input, and prints out on the terminal a pattern similar to the following. An example is shown for *n* = 6; note that your program should work for all inputs *n* between 1 and 9.

```
6
55
444
3333
22222
111111
```

4. [15 marks] Write a program that takes an integer *n* between 1 and 9 as input, and prints out on the terminal a pattern similar to the following (example shown for *n* = 5).

```
1        1
22      22
333    333
4444  4444
5555555555
```

Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20
Assignment 4 | January 04, 2022

Submission instructions

* Submit one .c file for each part of the assignment. Each of your .c files should be named as:

<your roll number>_A<assignment number>_<part number>.c

For instance, if your roll number is 21CS10023, and if you are presently doing Assignment 4 which has 4 parts, then you should submit 4 separate .c files named as:
21CS10023_A4_1.c   21CS10023_A4_2.c   21CS10023_A4_3.c   21CS10023_A4_4.c

* Submissions must be through the course Moodle, before the end of Lab session (11:55 AM). Late submissions will be penalized / not accepted.

---

1. [10 Marks] Write a program to separate odd and even integers in separate arrays. Assume that the array can hold maximum 10 integers.
input: No. of elements in the array, the different integer values in the array
output: One array for odd integers and one array for even integers
Example output:

```
Enter the number of elements: 5
Enter the elements separated by space: 4 8 3 4 6
Even Integers: 4 8 4 6
Odd Integers: 3
```

2. [10 Marks] Write a program that will insert a new value to an already existing sorted array such that the sorting is maintained. You can assume
1. The original array is sorted in ascending order
2. That the user provides the elements of the original array in already sorted order
3. That the array can hold maximum 10 values
input: No. of elements in the array, the different values in the array in sorted order, the new value to be inserted
output: The original array, the modified array
Example output:

```
Enter the num of elements (Max Size =9): 5
Enter the elements in sorted order (separated by space): 5 10 15 20
25
Enter the new value to be inserted: 17
Old array: 5 10 15 20 25
Modified array: 5 10 15 17 20 25

Enter the num of elements (Max Size =9): 5
Enter the elements in sorted order (separated by space): 5 10 15 20
25
Enter the new value to be inserted: 30
Old array: 5 10 15 20 25
Modified array: 5 10 15 20 25 30
```

3. [15 Marks] Fill a single dimensional integer array of size 30 with random integral numbers in the range **5** to **50** by calling the **rand()** library function. Please visit the following link [http://www.cplusplus.com/reference/cstdlib/rand/] to know how to use **rand()** function to generate random integers between a lower and upper bound. It should then display the array

contents nicely formatted. Then reverse the elements of the integer array, so that the last element becomes the first, the second from last becomes the second, and so on. It should reverse the elements in place – that is, without using another array. It should display the contents of the array after reversal.

Example output:

```
The original array is: 22 18 32 35 17 13 49 23 30 22 41 40 11 13 42
8 28 24 5 31 30 34 12 38 29 10 16 11 35 14
The reversed array is: 14 35 11 16 10 29 38 12 34 30 31 5 24 28 8 42
13 11 40 41 22 30 23 49 13 17 35 32 18 22
```

4. [15 Marks] Write a program that takes as input an integer array a[10] that stores n (= at most 10) elements (scanned during execution) and finds how many elements are out of order. An element a[i] is said to be in order if it is not smaller than a[0], a[1], ..., a[i-1] and not larger than a[i+1], a[i+2], ..., a[n-1].

Example output:

```
Enter number of elements: 2
Enter 2 integers (separated by space): 5 7
Not in order = 0

Enter number of elements: 4
Enter 4 integers (separated by space): 5 6 7 9
Not in order = 0

Enter number of elements: 4
Enter 4 integers (separated by space): 5 5 7 5
Not in order = 2

Enter number of elements: 7
Enter 7 integers (separated by space): 2 4 2 2 4 2 2
Not in order = 6
```

Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20
Assignment 5 | January 18, 2022

Submission instructions

* Submit one .c file for each part of the assignment. Each of your .c files should be named as:

<your roll number>_A<assignment number>_<part number>.c

For instance, if your roll number is 21CS10023, and if you are presently doing Assignment 5 which has 3 parts, then you should submit 4 separate .c files named as:
21CS10023_A5_1.c   21CS10023_A5_2.c   21CS10023_A5_3.c

* Submissions must be through the course Moodle, before the end of Lab session (11:55 AM). Late submissions will be penalized / not accepted.
* Try to think how you can solve a problem efficiently. A program that solves the problem efficiently will get more credit.

---

1. [15 Marks] Write a c program where two sorted arrays A, B are taken as input from the user in the main function. Write a function 'merge_array' that takes in two sorted arrays and merges those two arrays to get a merged sorted array. From the main function, invoke the 'merge_array' function, pass both the arrays to it. Print that merged sorted array from the main function. Next take another sorted array C as input in main. Again invoke the 'merge_array' function which takes the previously merged (sorted) array and array C as input and merges them to get a new sorted array. Print the finally sorted array from the main function. You can take the max size of each of the user supplied arrays to be 10 and assume the user supplies the number of elements of each array before supplying the array elements.
Example output:
```
Enter the number of elements in the first array (A): 3
Enter the elements of the first array (A): 4 7 12
Enter the number of elements in the second array (B): 4
Enter the elements of the second array (B): 5 6 7 10
Merged array: 4  5  6  7  7  10  12
Enter the number of elements in the third array (C): 4
Enter the elements of the third array (C): 2 3 7 8
Merged array: 2  3  4  5  6  7  7  7  8  10  12
```

2. [15 Marks] Write two functions "int reverse(int N)" and "int reverse_recursive(int N)" both of which takes a decimal integer N as argument and returns the integer which is formed by reversing the order of digits of N. In "int reverse(int N)", do not use recursion to get the reversed number while in "int reverse_recursive(int N)" recursively call the function to get the reversed number. If required, you can use a global variable [that can be used throughout the program]. Write a main() function to demonstrate the use of both the functions.
Example output:
```
Please provide the positive integer to be reversed: 1234
The reverse of 1234 using "reverse_recursive" is: 4321
The reverse of 1234 using "reverse" is: 4321

Please provide the positive integer to be reversed: 7890
The reverse of 7890 using "reverse_recursive" is: 987
The reverse of 7890 using "reverse" is: 987
```

3. [20 Marks] For any real number x, the series expansion of sin(x) is as follows:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Write a program that takes an user given real number x [where x (in degrees) is the value of the angle] and computes the value of sin(x) up to a precision of 0.0001, that is the actual value of sin(x) and the value of sin(x) computed by the above formula, should not differ by a factor of more than 0.0001. You should write a function "sine_series" which takes the angle value supplied by the user. The return type and the argument types have to be properly set. The function should compute the value of the sin of the angle using the formula as said above and return the computed value. Call the function from main and print the value of sin of x from main. You can use math.h library functions to get the actual value of sin(x) which you will need to compare to the computed value of sin(x) upto a precision of 0.0001.

Example output:

```
The value of x (in degrees): 45
The value of sin45.000000 is 0.707215

The value of x (in degrees): 30
The value of sin30.000000 is 0.500061
```

Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20
Assignment 6 | January 25, 2022

Submission instructions

* Submit one .c file for each part of the assignment. Each of your .c files should be named as:

<your roll number>_A<assignment number>_<part number>.c

For instance, if your roll number is 21CS10023, and if you are presently doing Assignment 6 which has 3 parts, then you should submit 3 separate .c files named as:
21CS10023_A6_1.c 21CS10023_A6_2.c 21CS10023_A6_3.c

* Submissions must be through the course Moodle, before the end of Lab session (11:55 AM). Late submissions will be penalized / not accepted.

---

1. **[20 marks]** Define a structure *comp* to store a complex number of the form A+iB, where A (the real part) and B (the imaginary part) are real numbers. Write the following functions:

- A function that takes a complex number as argument, and displays the complex number in the format A+iB or A-iB as appropriate
- A function that takes two complex numbers as arguments, and returns the sum (another complex number)
- A function that takes two complex numbers as arguments, and returns the product (another complex number)
- A function that takes a complex number and displays the number in polar form, i.e., displays the absolute value and argument (phase) of the complex number

Write a main function that takes complex numbers as inputs from the user, and demonstrates the working of the above functions.

You can refer to https://en.wikipedia.org/wiki/Complex_number

2. [15 marks] Design a structure *student* containing the data of a student. The fields of the structure will be roll_num, name, facad_code (all strings). Design another structure named *faculty*, containing the data of a faculty member, having the fields emp_code, name (both strings).

The facad_code field for a student *s* contains the emp_code of the faculty member who is the faculty advisor for the student s.

Declare arrays of the two structures, to store records of up to 10 students and 10 faculty members.

Write functions for the following operations:

(i) Given the roll_num of a student, print out the name of his / her faculty advisor.

(ii) Given the name of a faculty member, print out the roll numbers and names of ALL those students who are advised by the specified faculty member.

Write a main program that creates the records for 5 students and 3 faculty members and demonstrates the working of the above functions. A sample list of records is given below:

Students

Roll Number	**Name**	**FacAd Code**
21RB033	Max	16AL2014
21FR016	Charles	19AM2105
20MC003	Daniel	19AM2105
22MR063	George	18MS2144
22FR055	Carlos	16AL2014

Faculty

Employee code	**Name**
18MS2144	Lewis
16AL2014	Alonso
19AM2105	Sebastian

3. [15 marks] Write a program that takes as input from the user, a NxN square matrix of integers (which will be stored as a 2-d array). You can assume that N will be at most 10, hence you can declare the 2-d array accordingly. The main() function should take the elements of the matrix as input from the user, displaying suitable prompts, e.g.,

```
Enter the dimension of the square matrix (N): 3
Enter the value of element [1][1]: -4
Enter the value of element [1][2]: 3
...
Enter the value of element [3][3]: 9
```

Write a function that takes this 2-d array as argument, and computes (i) the largest element in the main diagonal, and (ii) the smallest element in the secondary diagonal. The function should inform these two values to the main() function (using pointers), and the main() function should print them out.

Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20
Assignment 7 | February 15, 2022

Submission instructions

* Submit one .c file for each part of the assignment. Each of your .c files should be named as:

<your roll number>_A<assignment number>_<part number>.c

For instance, if your roll number is 21CS10023, and if you are presently doing Assignment 7 which has 3 parts, then you should submit 3 separate .c files named as:
21CS10023_A7_1.c 21CS10023_A7_2.c 21CS10023_A7_3.c

* Submissions must be through the course Moodle, before the end of Lab session (11:55 AM). Late submissions will be penalized / not accepted.

---

1. **[15 marks]** Take two strings s1 and s2 as inputs from the user. Assume that s1 and s2 are sequences of English words. Combine s1 and s2 into a strings s of words as follows: Insert words alternately from s1 and s2 in s, starting with s1. If all the words of one of the strings s1 or s2 have already been inserted into s, copy the rest of the other string into s. Finally print s on the screen.

Example interaction with user:

```
Enter s1: It is raining outside

Enter s2: Ram has gone to market

It Ram is has raining gone outside to market
```

2. **[15 marks]** Write a C program that stores the following 10 strings using a 2-dimensional character array: “messi”, “ronaldo”, “beckham”, “romario”, “klose”, “ronaldinho”, “kaka”, “maradona”, “pele”, and “zidane”. Each row of the 2-d array should store one string.

Then the program should ask the user for inputing one character. The program should search for the input character in all strings, and print out that string which contains the input character most number of times. For instance, if the input character is ‘s’, the program should print out “messi”; if the input is ‘k’, the program should print out “kaka”. In case there are multiple strings which contain the input character the maximum number of times (i.e., in case of a tie), then all those strings should be printed out, one in each line.

3. **[20 marks]** Write a program that helps a user to maintain a dynamic set of integers. The program should start with an empty integer array of size 100, which will be used to store the set. The user should be given options to (i) insert numbers from this set, (ii) delete numbers from this set, (iii) display the set in set notation (i.e., within curly braces, and elements separated by comma), and (iv) exit from the program. When the user wants to insert a number, the program should check whether the number already exists in the set (every element of a set should be distinct). If yes, the program should inform the user; otherwise the element should be inserted into the array. Again, when the user wants to delete

a number, the program should check whether the number exists in the set. If not, the user should be informed; otherwise the said number should be deleted from the set.

There should be separate functions for inserting a number into the set, deleting a number from the set, and for displaying the set.

Throughout the operations, the array should be maintained sorted in ascending order, and binary searching should be used to check for the presence of numbers. It will be helpful if you write a search function (using binary search) to detect if a certain element is contained in the set; this function can be used both within the insert and delete functions.

An example interaction between the user (U) and the program (P) is as follows:

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 1

P: Enter number to insert:

U: 43

P: 43 inserted [Note: 43 should be placed in the first element of the array]

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 1

P: Enter number to insert:

U: 21

P: 21 inserted [Note: 21 should be placed in the first element of the array, and 43 should be shifted to the second position.]

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 1

P: Enter number to insert:

U: 73

P: 73 inserted [Note: 73 should be placed in the third element of the array]

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 1

P: Enter number to insert:

U: 43

P: Number already exists

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 3

P: {21, 43, 73}

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 2

P: Enter number to delete

U: 43

P: 43 deleted [Note: 73 should be brought to the second element of the array]

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 2

P: Enter number to delete

U: 50

P: This number does not exist in the set

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 0

Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20
Assignment 8 | February 22, 2022

Submission instructions

* Submit one .c file for each part of the assignment. Each of your .c files should be named as:

<your roll number>_A<assignment number>_<part number>.c

For instance, if your roll number is 21CS10023, and if you are presently doing Assignment 8 which has 3 parts, then you should submit 3 separate .c files named as:
21CS10023_A8_1.c   21CS10023_A8_2.c   21CS10023_A8_3.c

* Submissions must be through the course Moodle, before the end of Lab session (11:55 AM). Late submissions will be penalized / not accepted.
* Try to think how you can solve a problem efficiently. A program that solves the problem efficiently will get more credit.

---

1. [10 Marks] Declare an integer variable `n`. Read its value from the keyboard and dynamically allocate an integer array `a[]` of length `n`. Fill up the array by `n` integers value from the keyboard. Print the elements of `a[]` in the terminal.
Write a function that takes the address and the length of an array as arguments, and shifts its content to the right by one place with wrap around, as shown in the example. Call this function from the `main()` with `a[]` and the length of the array as input. Print the array `a[]` in the terminal after the function call. Do this one more time.
Example output:

```
Enter number of elements: 5
Enter the elements: 7 3 6 1 4
a[] = 7 3 6 1 4
After function call:
a[] = 4 7 3 6 1
After function call:
a[] = 1 4 7 3 6
```

2. [20 Marks] Declare a 2D integer array a[][] and define its size (#rows and #columns) taking user input. Write a function to dynamically allocate the space for a[][]. Fill it up by integer values from the user. Write another function to find its saddle point(s) and print on the terminal. If there is none, then print that message.
An element is a saddle point if it is the minimum in its row as well as the maximum in its column. Similarly, if an element is the maximum in its row as well as the minimum in its column, then also it is a saddle point.
Assume that the user enters the matrix such that the minimum or maximum values in any row or column is **unique**. That is – the minimum or maximum value does not repeat inside the same row or column.
Example output:

```
Enter number rows and columns: 3 3
Enter the array (one row at a time):
3 1 7
4 3 8
1 2 5
Saddle point is a[1][1]=3
Saddle point is a[2][2]=5

Enter number rows and columns: 5 5
Enter the array (one row at a time):
```

```
23 25 19 55 11
80 50 69 55 70
11 15 45 95 19
24 25 33 50 32
62 35 22 78 35
Saddle point is a[1][1]=50
Saddle point is a[3][3]=50

Enter number rows and columns: 3 3
Enter the array (one row at a time):
2 5 7
1 2 5
7 3 4
No saddle point exists for the given input matrix.
```

3. [20 Marks] "Favorite books" in a library are those which have been read by all those members who have read at least 50% books. For this, consider a library with m members and b books. The values of m and b are given as input. Moreover, the library also maintains an integer matrix S where S[i][j] = 1 if the i-th member has read the j-th book, and S[i][j] = 0 otherwise. Dynamically allocate the m-by-b 2D array S and fill it up using data from the user.
(a) Find and print the members who have read at least 50% books.
(b) Find and print the favourite books.
Write appropriate functions to solve the problem.
Hint: Try to resuse some part of the programs you wrote in the previous problem.

Example:

```
#members = 3. #books = 5.
S matrix:
1 1 1 0 0
0 0 1 1 0
0 0 0 0 1
Members reading at least 3 books: 1.
And the favorite books are: 1, 2, 3.

#members = 7. #books = 12.
S matrix:
1 1 0 0 0 1 0 0 0 0 0 1
0 0 1 1 1 0 1 1 0 0 0 0
1 1 1 1 0 1 0 1 1 1 0 1
0 0 0 1 1 1 0 0 0 0 0 0
1 1 1 1 0 0 0 1 0 0 1 0
0 1 1 1 0 1 0 0 0 0 0 1
0 1 1 0 1 0 1 1 0 0 1 1
Members reading at least 6 books: 3, 5, 7.
And the favorite books are: 2, 3, 8.
```

Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20
Assignment 9 | March 08, 2022

Submission instructions

* Submit one .c file for each part of the assignment. Each of your .c files should be named as:

<your roll number>_A<assignment number>_<part number>.c

For instance, if your roll number is 21CS10023, and if you are presently doing Assignment 9 which has 2 parts, then you should submit 2 separate .c files named as:
21CS10023_A9_1.c 21CS10023_A9_2.c

* Submissions must be through the course Moodle, before the end of Lab session (11:55 AM). Late submissions will be penalized / not accepted.
* Try to think how you can solve a problem efficiently. A program that solves the problem efficiently will get more credit.

---

1. [25 Marks] Write a function to reverse a linked list with 5 values [entered by the user], where the value at each node is an integer. While you create the linked list by taking input from the user, assume you insert the elements at the end of the list. Don't just print in reverse order. You have to actually reverse the original linked list, whereby the last node will become the head node.
Example output:
```
Enter number 0: 11
Enter number 1: 12
Enter number 2: 13
Enter number 3: 14
Enter number 4: 15

Original List: 11 12 13 14 15
Reversed List: 15 14 13 12 11
```

2. [25 Marks] Declare globally a structure named "stu" that has the following attributes:
name of a student (character string);
marks (integer).
Take as user's input the number of students and their names and marks in non-decreasing order of marks. Dynamically allocate a 1D array a[] and store these records in a[].
Write a function named "binary_search(...)" to search the record with the smallest index j such that the marks in a[j], a[j+1], ..., a[n-1] are no less than some cut-off marks (user input). This function is not exactly binary search taught in the class but a modified version. Print the records a[j], a[j+1], ..., a[n-1].
Example output:
```
Enter #students: 3
Enter the names and marks:
Bhanu      17
Vishnu     19
Aayush     21

Enter the cut-off marks: 15

Bhanu      17
Vishnu     19
Aayush     21
```

Enter #students: 3

```
Enter the names and marks:
Bhanu      17
Vishnu     19
Aayush     21

Enter the cut-off marks: 19

Vishnu     19
Aayush     21
----------------------------------
Enter #students: 3
Enter the names and marks:
Bhanu      17
Vishnu     19
Aayush     21

Enter the cut-off marks: 21

Aayush     21
----------------------------------
Enter #students: 3
Enter the names and marks:
Bhanu      17
Vishnu     19
Aayush     21

Enter the cut-off marks: 25

Empty!
----------------------------------
Enter #students: 9
Enter the names and marks:
Bhanu      17
Vishnu     19
Aayush     21
Raj        35
Samboji    39
Shikha     40
Satyam     40
Hardik     42
Vaibhav    49

Enter the cut-off marks: 40

Students with at least cut-off marks:
Shikha     40
Satyam     40
Hardik     42
Vaibhav    49
```

Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20

Lab Test 1 | January 11, 2022
Full marks: 3 x 20 = 60
Time: 09:15 – 11:45 (2 hours 30 minutes)

Instructions

* There are three (3) problems in this test, each of 20 marks. The program for each problem must be written in an individual C source file. You should submit the following three plaintext C source files: **<roll number>_T1_1.c, <roll number>_T1_2.c,** and **<roll number>_T1_3.c**

* Submission must be through the course Moodle, before 11:45 (according to the Moodle clock). If you miss this deadline, then you need to email your submission to TA Anurag Roy (Email: anurag_roy@iitkgp.ac.in) within 12:00; there will be a penalty of 30 marks for such late submissions. No submission will be allowed after 12:00; any submission reaching the TA's mailbox after 12:00 (according to the receipt timestamp of the email) will be rejected.

* You are NOT supposed to take the help of any person / TA / book / online material during the test. Any malpractice / plagiarism will be penalised severely, with the minimum being awarding zero for the entire test.

* It is your responsibility to make your programs understandable, through meaningful variable names, indentation, comments (if necessary). Programs that are not understandable will be penalized.

Problem 1

Write a program that takes an integer *n* between 1 and 9 as input, and prints out on the screen a pattern similar to the following. An example is shown for *n* = 5; note that your program should work for all inputs *n* between 1 and 9.

```
55555
 4444
  333
   22
    1
   22
  333
 4444
55555
```

Problem 2

Given a set of distinct positive integers (i.e., no duplicate values), you need to find the subsets of the given set such that each subset contains exactly three consecutive integers. For instance:

- The set {100, 56, 5, 6, 102, 58, 101, 57, 7} has 3 such subsets {5, 6, 7}, {100, 101, 102} and {56, 57, 58}.
- The set {100, 56, 6, 102, 58, 101, 57, 7, 103} has 3 such subsets {100, 101, 102}, {101, 102, 103} and {56, 57, 58}.
- The set {100, 56, 104, 6, 101, 57, 7, 103} does not have any such subset.

Write a C program that asks the user to enter a set of numbers and stores the numbers in an array, and then prints the subsets containing exactly three consecutive integers. You can assume that the input set will contain at most 100 integers, and that the user will enter distinct, positive integers (no need to check for duplicates).

Problem 3

Consider two arrays A and B which store *n* and *m* (user inputs) integers respectively. Both arrays can contain positive and negative integers. Write a program that first asks the user the values of *n* and *m* (i.e., the number of elements in the two arrays) and then takes *n* and *m* integers as input from the user, and stores in the two arrays. You can assume that maximum value of *n* and *m* will be 50.

Consider another integer array C that is initially empty. Your program should copy values from the arrays A and B to array C in the following manner:
(i) Start reading the elements in A and copy the elements to C (in the same order), as long as there is no change in the sign of the elements in A.
(ii) Once there is a sign-change, start reading elements from B, and copy those to C (in the same order) as long as there is no change in the sign of the elements in B; once there is a sign-change in the elements of B, again resume with the elements in A.
(iii) Continue the above process till one or both of the arrays A and B is exhausted. Copy the rest of the elements of the other array (if any) into C. Your program should print out the array C.

Example interaction of the program with user:

```
Enter number of elements in array A: 7
Enter elements in array A: 1 2 3 -4 -5 6 7
Enter number of elements in array B: 5
Enter elements in array B: 11 12 -13 -14 -15
Array C: 1 2 3 11 12 -4 -5 -13 -14 -15  6  7

Enter number of elements in array A: 7
Enter elements in array A: 10  -6  12  13  -67  32  6
Enter number of elements in array B: 5
Enter elements in array B: 14  7  -12  32  -8
Array C: 10  14  7  -6  -12  12  13  32  -67  -8  32  6
```

Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20
Lab Test 2 | February 01, 2022
Full marks: 3 x 20 = 60
Time: 09:15 – 11:45 am (2 hours 30 minutes)

Instructions

* There are three (3) problems in this test, each of 20 marks. The program for each problem must be written in an individual C source file. You should submit the following three plain text C source files: **<rollnumber>_T2_1.c, <rollnumber>_T2_2.c and <rollnumber>_T2_3.c** .

* Submissions must be through the course Moodle, before **11:45 AM** (according to the Moodle clock). If you miss this deadline, then you need to email your submission to TA Owais Iqbal (Email: owais.iqbal@kgpian.iitkgp.ac.in) within 12:00 noon; there will be a penalty of 30 % of marks for such late submissions. No submission will be allowed after 12:00 noon; any submission reaching the TA's mailbox after 12:00 (according to the receipt timestamp of the email) will be rejected.
* You are NOT supposed to take the help of any person/TA/book/online material during the test. Any malpractice/plagiarism will be penalised severely, with the minimum being awarding zero for the entire test.
* It is your responsibility to make your programs understandable, through meaningful variable names, indentation, comments (if necessary). Programs that are not understandable will be penalized.

---

1. [20 Marks] Define a structure: `struct point{float x, y;};` Write a function that takes two points as input and returns their Euclidian distance to the calling function. Its prototype will be: `float calculate_distance(struct point p, struct point q).`
Declare an array of such points in `main()`. You can take the maximum size of the array as 10. From the user take `n` as the number of points to be entered, fill up the array with `n` points. Find the minimum distance among these `n` points by calling the function `calculate_distance` from `main()` and print this distance.
<u>Example output</u>:

```
Enter no. of points (2 to 10): 3
Enter (x,y) coordinates of the points:
0 0
3 3
2 2
Minimum inter-point distance = 1.414214

Enter no. of points (2 to 10): 5
Enter (x,y) coordinates of the points:
11.11 22.22
22.22 33.33
33.33 44.44
44.44 55.55
55.55 66.66
Minimum inter-point distance = 15.711910
```

2. [20 Marks] Write a function that takes as parameters two numbers x (real) and y (positive integer) and returns the value of x + x^2 + ... + x^y. Call this function from `main()` and print the value of the expression **from `main()`**. Note the examples below. You should print both the series expression as well as the value of the series expression. You are **not allowed** to use `math.h` header file.
<u>Example output</u>:

```
Enter x: 4
Enter y: 3
4.000000 + 4.000000^2 + 4.000000^3 is: 84.000000

Enter x: 1.5
Enter y: 4
1.500000 + 1.500000^2 + 1.500000^3 + 1.500000^4 is: 12.187500
```

3. [20 Marks] Write a C program that should define a **square** matrix in the main function. In the main function call the following functions to perform some matrix operations. The maximum number of rows or columns can be taken as 4.

   a. [3 marks] **void matrixInput(float mat[][N])**: This function should fill the 2D array with user supplied values by appropriately prompting the user. Display the array nicely formatted (i.e., in row-column format).

   b. [7 Marks] **void matTranspose (float A[][N], float AT[][N])**: This function should first display the matrix A nicely formatted. Then it should transpose the matrix A and store the transposed matrix in AT and display the transposed matrix.

   c. [10 Marks] **void OrthoNormal(float A[][N], float AT[][N])**: This function should first display the matrix A nicely formatted. A square N X N matrix "A" is said to be orthonormal if A*AT = I where AT is the transpose of the matrix A and I is the NXN identity matrix. Determine whether the matrix A is orthonormal or not and display an appropriate message.

<u>Example of an Orthonormal Matrix is</u>:

```
0.00 -0.80 -0.60
0.80 -0.36 0.48
0.60 0.48 -0.64
```

<u>Example of a matrix which is not Orthonormal is</u>:

```
1.00 2.00
3.00 4.00
```