Assignment 1

Write a C Program that reads in three points on a 2 dimensional plane (integers). All three will not be in a single straight line. These are corner points of an isosceles triangle T1. You will first test whether the triangle T1 is indeed an isosceles triangle or not. If it is isosceles, you will find the lengths of base, height and also the area of the triangle T1 and print them. Next, you will find the length of the isosceles side of a Triangle T2, which has double the area and three times the base length as T1 and print it.

Save the file as A01_<Roll Number>.c (example A01_21AG10002.c). Build, Run for and test it for the given data. Then upload the .c file for the Assignment.

In particular, you will do the following:

1. Read x and y integer coordinates of three points of T1 and print them.
2. Find out the lengths of the three sides of T1 and print them.
3. Test whether they form an isosceles triangle. If no, print that T1 is not an isosceles triangle and end.
4. If it is an isosceles triangle, then print the lengths of base and height of T1 and also print the area of T1.
5. If T1 is an isosceles triangle, then print the length of the isosceles side of another isosceles triangle T2 whose base is three times that of T1 and area is twice that of T1.

Test Data:

Test 1: <0,0>, <10,0>, <5,5>, Test 2: <4,12>, <8,16>, <3,3>

Test 3: <10,10>, <-3,3>, <3,-3>

Assignment 2

Write a C Program that will read four integer values, find how may of them are unique and print only the unique elements in ascending order. For example, if the integers read in are 5, 8, 1,5 then you will first print that there are 3 unique elements and then print 1, 5, 8.

Save the file as A02_<Roll Number>.c (example A02_21AG10002.c). Build, Run for and test it for the given data. Then upload the .c file for the Assignment.

In particular, you will do the following:

1. Read in four integers and print them.
2. Print how many unique elements are there.
3. Print the unique elements in ascending order.

Test Data:

1. 7, 1, 7, 1
2. 4, 4, 4, 2
3. 1, 5, 8, 2
4. 5, 5, 5, 5

Assignment 3 (Loops or Arrays are NOT to be Used)

Write a C Program that reads in five integers. If the numbers are not distinct, it prints "NOT DISTINCT" and terminates. If they are all distinct, then it finds and prints the following: largest, smallest, sum and median of the five numbers.

Save the file as A03_<Roll Number>.c (example A03_21AG10002.c). Build, Run for and test it for the given data. Then upload the .c file for the Assignment.

In particular, you will do the following:

1. Read five integers and print them. They must be printed in a single line with commas in between numbers.
2. If the numbers are not distinct, in a new line, print "NOT DISTINCT" and terminate
3. If they are all distinct, then print the following in a new line each
   a) Largest of the five numbers as – Largest = <Print the largest number>
   b) Smallest of the five numbers as – Smallest = <Print the smallest number>
   c) Sum of the five numbers as – Sum = <Print the sum>
   d) Median of the five numbers as – Median = <Median of the five numbers>

Test Data: (Inputs will be given one after another without commas)

a) 7 9 23 1 5
b) 3 7 19 -1 8
c) 4 1 9 11 9
d) -2 8 0 -1 -9

Assignment 4 (Loop CAN be used but NOT arrays)

Write a C Program that will read in a set of at most 10 positive integers. However, if an integer 0 or less is read somewhere, then that means that no more integers are to be read. Now among the positive integers read, find out if the sum of those which are divisible by either 2 or 3 or 5 is greater than the sum of those which are not divisible by any one of 2 or 3 or 5, or whether the two sums are equal. (Loop CAN be used and will help to write a shorter program, though it is not absolutely necessary to use loops. Arrays are NOT to be used)

Save the file as A04_<Roll Number>.c (example A04_21AG10002.c). Build, Run for and test it for the given data. Then upload the .c file for the Assignment.

In particular, you will do the following:

1. Read in integers one by one and print them till either you read 10 positive numbers or read a 0 or negative number. (Print only the positive numbers). Create a new line after they are printed.
2. In a new line, Print the sum (Sum1) of numbers divisible by any one of 2 or 3 or 5 as – Sum1 (Divisible) = <Value>
3. In a new line, Print the sum (Sum2) of numbers not divisible by any of 2 or 3 or 5 as – Sum2 (Not Divisible) = <Value>
4. In a new line, Print which sum is larger or whether they are equal.

Test Data: (Inputs will be given one after another without commas)

a) 2 3 5 7 15 11 7 0

b) 8 12 23 17 19 18 -4

c) 11 23 18 16 29 31 9 12 16 7

Additional Assignment 1, Dec 22, 2021

Write a Program to read in 5 positive integers and find the largest common digit among them. If there is no common digit, then print no. DO NOT USE ARRAYS.

Assignment 5 (Arrays are NOT to be Used)

Write a C Program to evaluate the series

$f(x) = 1 - 2x/1! + 3x^2/2! - 4x^3/3! + 5x^4/4! - \ldots$

for $0 < x \leq 1$ (x is a floating point number)

up to a given integer number n (n > 0) of terms.

Save the file as A05_<Roll Number>.c (example A05_21AG10002.c). Build, Run for and test it for the given data as well as your own data. Then upload the .c file for the Assignment.

[20 Marks: 5 marks for intermediate submission, 15 marks for final submission]

In particular, you will do the following:

1. Read in the value of x and an integer n and print them in a single line depicting which is x and which is n
2. If any one of the values entered is invalid, that is, either x does not satisfy $0 < x \leq 1$, or n is ≤ 0, print a message saying invalid input and ask the user to enter BOTH the values again, and read the values again (must use a while loop) and print them
3. Continue the above two steps until values entered satisfy both $0 < x \leq 1$ and n is positive
4. Now compute the value of f(x) up to n terms. Print the interim values of sum and term in every iteration inside the loop and then the final values at the end. Do not use any mathematical function like pow, etc.

Test Data:

a) 0.6 10

b) 0.97 25

c) 1.3 12 followed by 0.5 0 followed by 1.0 20

Assignment 6 (Arrays are NOT to be used)

Write a C Program that reads in a positive integer n, n > 100, and then finds all three digit numbers formed by consecutive digits of n. It then finds the largest prime number among these three digit numbers and prints it. It prints "None is Prime" if no such primes exist.

For example, if the number read is 5713271 then the program will print 571, 713, 132, 327, 271 and then say that the largest prime among them is 571. Again if the number read is 668844 then it prints 668, 688, 884, 844 and then prints "None is Prime"

Save the file as A06_<Roll Number>.c (example A06_21AG10002.c). Build, Run for and test it for the given data. Then upload the .c file for the Assignment.

[20 Marks: 5 marks for intermediate submission, 15 marks for final submission]

In particular, you will do the following:

1. Read in an integer n. (Assume a valid input will be given.) Print it. Create a new line after the print statement.
2. Print all three digit numbers of n as required above, one after another, in the same line, each number followed by a comma. After that, when all are printed, make a new line.
3. Print the largest primes number among these three digit numbers or state "None is Prime". (Write the code for checking for primes in the main program itself and do not write a separate function for detection of primes)

Test Data:

a) 5713271

b) 6688441

c) 1122335

Assignment 7 (Arrays are to be Used)

Write a C Program to read in n (assume 0 < n <=20) positive integers and then find out all non-duplicate pairs whose sum of odd valued digits have a difference between 5 and 20 but that difference is not divisible by 3.

For example, if n = 4 and the numbers are 3456, 1237, 8888, 77777, then the sum of odd valued digits are 8 (for 3456), 11 (for 1237), 0 (for 8888) and 35 (for 77777) and the pairs are (3456, 8888) and (1237, 8888). The other pairs do not satisfy the condition. [You may print the pairs in any order].

Save the file as A07_<Roll Number>.c (example A07_21AG10002.c). Build, Run and test it for the given data as well as your own data. Then upload the .c file for the Assignment.

[20 Marks: 5 marks for intermediate submission, 15 marks for final submission]

In particular, you will do the following:

1. Read in a positive integer n (0 < n <= 20) and print it. Create a new line.
2. Read in n positive integers and store them in an array. Print them in a single line followed by commas. Then create a new line.
3. Store the sum of odd-valued digits of every number in a separate array. Print each number along with the sum of odd digits of that number appropriately.
4. Print all pairs of numbers whose sum of odd valued digits have a difference between 5 and 20 but that difference is not divisible by 3. Do not print duplicate pairs. If there is no such pair, print “No Such Pair”

Test Data:

a) n = 5, numbers are 3456, 1237, 8888, 77777, 7321

b) n = 5, numbers are 11111, 23456, 1134, 99999, 77777

c) n = 4, numbers are 351, 531, 1345, 68153

Assignment 8 (Arrays are to be used)

Write a C Program that reads in a string of characters (without blanks in between) and then prints only the unique alphabets (replacing all upper case elements by their lower case) as a string which has the vowels first followed by the consonants after all the vowels. No numeric or other non-alphabet is printed. If there is no alphabet, print “No Alphabet in String”.

For example, if the string read is IITKharagpur$India then the program will first print IITKharagpur$India and then print iautkhrgpnd. (It could also print the result string in any sequence, like aiutkdhgrnp, etc).

Save the file as A08_<Roll Number>.c (example A08_21AG10002.c). Build, Run and test it for the given data. Then upload the .c file for the Assignment.

[20 Marks: 5 marks for intermediate submission, 15 marks for final submission]

In particular, you will do the following:

1. Read in a character string in %s format and store it in a character array. Print the string read in and create a new line.
2. In a separate array, create a string formed by only the unique alphabets of the original string, replacing all upper case characters by their lower case, removing all non-alphabets and having the vowels before the consonants.
3. Print this new string in a suitable fashion and create a new line. If there is no alphabet, print “No Alphabet in String”. (Do not use any special string library function except to read and write in %s format.) [If you wish, you can also use an intermediate third array for your work, but that is not necessary]

Test Data:

a) IITKharagpur$India
b) PDSLabSection11
c) A*Algorithm#AI
d) &*5123!

Assignment 9 (User Defined Functions are to be Used)

A positive integer is said to have an increasing digit k-sequence (of length k), if the first k digits are in strictly increasing value sequence. For example, the k-sequence of 58345 is 2, 23567 is 5, 71289 is 1, 45667 is 3. Write a non-recursive function seq(n) which takes an integer n and returns the increasing digit sequence number of n. Then in the main program, you should read in a positive integer m, read m integers, calculate the k-sequence of each integer you have read by calling the function seq(n) and print the results obtained. Also identify which of these integers are completely in increasing digit sequence (e.g., 23567).

Save the file as A09_<Roll Number>.c (example A09_21AG10002.c). Build, Run and Test it for the given data as well as your own data. Then upload the .c file for the Assignment.

[20 Marks: 5 marks for intermediate submission, 15 marks for final submission]

In particular, you will do the following:

1. Read in a positive integer m and print it suitably. Create a new line.
2. Read in m positive integers, repeatedly call the function seq (which you will define) for each of the m numbers that are read. Print each number and its k-sequence value appropriately in a new line each.
3. Among the above integers read, suitably print all those which are completely in increasing digit sequence. If there are no such, print “None have such a sequence”.
4. [Bonus] If you complete these within the scheduled time, write the function seq in a recursive manner. Call it rseq and use appropriate parameters. Test it like you tested seq. [Bonus will not add to assignment marks]

[Do not call any Library Functions other than scanf, printf. You may use your own user defined functions]

Test Data:

a) m = 5, numbers are 3456, 1237, 8888, 787, 7321

b) m = 5, numbers are 1, 23456, 134, 56789, 7897

c) m = 4, numbers are 351, 531, 1345, 68153

Assignment 10 (User Defined Functions are to be Used)

A string S is said to have a palindrome substring of length k (k must be > 1) if there is a substring P of length k in S which is a palindrome. For example, the string "abababc" has two palindrome substrings of length 5 ("babab" and "ababa"). It also has several palindrome substrings of length 3 (eg., "aba", etc). You are to read a string S and find the largest k (k>1) such that there is a palindrome substring P of length k in S, and print P and k. (If there are multiple such palindrome substrings of length k, print all of them). For this, you will write a non-recursive function pal(S[],i,j) which takes as arguments address of a string array s and integer indices i and j and returns 1 if the substring formed by the i_th to j_th consecutive characters of S form a palindrome. Otherwise it returns 0. For example if S "abababc", the call pal(S,2,6) will return 1 (since "babab" is a palindrome) while pal(S,1,4) will return 0 (since "abab" is not a palindrome). This function will be repeatedly called in the main program to solve the problem. [Do not distinguish between upper and lower case]

Save the file as A10_<Roll Number>.c (example A10_21AG10002.c). Build, Run and Test it for the given data as well as your own data. Then upload the .c file for the Assignment.

[20 Marks: 5 marks for intermediate submission, 15 marks for final submission]

In particular, you will do the following:

1. Read in a string S and print it. Create a new line.
2. Define and call the function pal appropriately to find all palindrome substrings of length k > 1 in S. Print each of them along with their i, j indices.
3. Find the largest palindrome substring of k > 1 in s and print all of them. If no such, print "No such".
4. [Bonus] If you complete these within the scheduled time, write the function pal in a recursive manner. Call it rpal and use appropriate parameters. Test it like you test pal. [Bonus will not add to assignment marks]

[Do not call any Library Functions other than scanf, printf. You may use your own user defined functions]

Test Data: abababababc, Malayalam, Bengal, Tatlallay.

Assignment 11 (Recursion)

You are to read in a positive integer n (n <= 20) and read n distinct positive integers into an array A. After that you are to recursively reverse the digits of each number of A and store them a separate array B. You will then sort both the arrays recursively, the original one (A) in ascending and one with reversed digits (B) in descending order. Then you will find the median of the combined list of elements of both arrays (that is A and B). For this you will write appropriate recursive functions for (a) reversing a digit (b) sorting in ascending / descending order, with an argument for whether it is to be in ascending or descending order, (c) finding the median of 2 sorted arrays. You will print all intermediate parts appropriately.

Save the file as A11_<Roll Number>.c (example A11_21AG10002.c). Build, Run and Test it for the given data as well as your own data. Then upload the .c file for the Assignment.

[20 Marks: 5 marks for intermediate submission, 15 marks for final submission]

In particular, you will do the following:

1. Read in a positive integer n and print it. Create a new line. Read in n positive integers in an array A and print them.
2. Reverse the digits of each array element of A to create a new number and store these in another array B. You will write a recursive function revdig(n) that reads an integer and returns a number with digits reversed (eg. 785 returns 587, 890 returns 98, 5 returns 5). You will call function revdig in the main program to reverse all the elements of A and store them in B. Print the elements of the array B.
3. Write a recursive sorting function rsort(....) with suitable arguments that you feel appropriate that takes an array and sorts it in ascending or descending order based on a binary variable called mode which is one of the parameters of rsort. You will call function rsort appropriately twice in the main program, once to sort A in ascending order (mode = 0) and then to sort B in descending order (mode = 1) and print these arrays A and B after sorting.
4. Write a recursive function rmed(....) with suitable arguments that you feel appropriate to find the median of all the elements of A and B. Call it from the main program and print the final median.

[If you use non-recursive functions for any part where recursive functions are to be used, then the maximum you can get for that part will 25% of the marks allotted for it. If you do not write any function for some part, you will get a 0 for that part.]

[Do not call any Library Functions other than scanf, printf. You may use your own user defined functions.]

Test Data: Test it with your own data

Assignment 12 (2-D Arrays, Functions, Recursion)

Read a positive integer n (n <= 20) and print it. Read the elements of an n by n 2-D matrix in which every element is distinct. Print the array elements neatly row-wise. Then read in a positive integer k and find the k-th largest element in the matrix (without moving the matrix elements from their locations) and print the k-th largest element along with its location index [i,j]. Next, you will sort all the elements of each row separately in ascending order and print the array column-wise (that is first printed line is column 1, 2nd printed line is column 2 and so on) after all rows are sorted. Now you will find the new location of the k-th largest element [p,q] and print it.

[It is recommended that you write your own functions for k-th largest and sorting, most preferably recursive functions. If you do not use your own functions you will get at most 10 marks if your code is correct. If you use correct non-recursive functions for finding k-th largest and sorting, you will get 3 additional marks for each and so a possible maximum of 16. If you write correct recursive functions for finding k-th largest and sorting, you will get 5 additional marks each and so a possible maximum of 20.]

Save the file as A12_<Roll Number>.c (example A12_21AG10002.c). Build, Run and Test it for the given data as well as your own data. Then upload the .c file for the Assignment.

[20 Marks: 5 marks for intermediate submission, 15 marks for final submission]

In particular, you will do the following:

1. Read in a positive integer n and print it suitably. Create a new line.
2. Read in an n by n matrix into a 2-D array and print it row-wise. Create a new line.
3. Read in an integer k and find the k-th largest element of the matrix using a function largek(....) with appropriate arguments, and print it along with its location index. If k is more than the number of elements of the matrix or less than 1, print a suitable message.
4. Sort each row of the array separately using a function sortr(...) with appropriate arguments, in ascending order and print this version in a column-wise manner.
5. Find the new location of the k-th largest element in the updated array and print the location index.

[Do not call any Library Functions other than scanf, printf. You may use your own user defined functions]

Test Data: Test it with your own data

Assignment 13 (Structures, Pointers, Functions)

You are to define data types for storing the following in a 2-dimensional plane: (a) POINT which takes x and y coordinates, (b) LINE which takes two end points, (c) CIRCLE which takes centre-point and value of radius and (d) axis-parallel RECTANGLE (that means its sides are parallel / perpendicular to the x-y axis) for which you take the left bottom and right top corner points. In the main program, you will appropriately read in a line L, a circle C and a rectangle R. You will then find out whether any two of these pairs (L-C, L-R, C-R) intersect each other and if so, at how many places along with the specific points of intersection. For this, you will write three appropriate functions for finding the answers to these questions for various pairs of intersections. Since these functions will need to return more than one item, appropriate use of structures and pointers may be made, as needed.

Save the file as A13_<Roll Number>.c (example A13_21AG10002.c). Build, Run and Test it for the given data as well as your own data. Then upload the .c file for the Assignment.

[20 Marks: 5 marks for intermediate submission, 15 marks for final submission]

In particular, you will do the following:

1. Define data types for POINT, LINE, CIRCLE and axis-parallel RECTANGLE. Use floating point.
2. Write three functions to find out various intersections between pairs of any two of the following: lines, circles and rectangles (that is, line-circle, line-rectangle, circle-rectangle). These functions will return whether and how many places they intersect, and if they intersect list the points of intersection.
3. In the main program read in a line L, a circle C, a rectangle R. You will then call the appropriate functions for the three possible combinations to return the appropriate results. The results will be printed in a suitable and readable form in the main program and not from within the functions.

[Do not call any Library Functions other than scanf, printf. You may use your own user defined non-recursive functions.]

Test Data: Test it with your own data for various cases.

Assignment 14 (Pointers, Dynamic Allocation, Functions)

You will read in positive integers m and n. Then you will dynamically allocate two strings A and B of sizes m and n, respectively. You will read the two strings A and B, each having lower case alphabets only. You will write three separate functions to find the following: The number of unique characters in their union, intersection, difference. These functions will be called for A and B, and the results returned to the main program, from where they will be printed neatly in a suitably understandable form. Once this is done, you will write another set of three functions to return the respective strings of their union, intersection and difference. For this you will dynamically declare the character arrays of the results based on the sizes of m and n. These functions will be called for A and B and the results returned to the main program. These returned results will be printed neatly in a suitably understandable form in the main program.

Save the file as A14_<Roll Number>.c (example A14_21AG10002.c). Build, Run and Test it for the given data as well as your own data. Then upload the .c file for the Assignment.

[20 Marks: 5 marks for intermediate submission, 15 marks for final submission]

In particular, you will do the following:

1. Read positive integers m and n and dynamically allocate strings A and B of sizes m and n. Take as input strings A and B and print them. Assume valid inputs will be provided.
2. Write functions to find the number of unique letters in union, intersection and difference of two strings and return the same. Call these in the main program for the various combinations of A and B, as respective arguments and print the results suitably from the main program.
3. Write functions to find the actual unique letters in the union, intersection and difference of two strings and return the same as a string pointer. Dynamically allocate the resulting string at the places needed, so that the result can be sent and received through the function calls. Call these functions in the main program for the various combinations of A and B and print the results suitably from the main program.

[Do not call any Library Functions other than scanf, printf. You may use your own user defined functions. They may be non-recursive or recursive, as per your choice]

Test Data: Test it with your own data for various cases.

Assignment 15

Read in a sentence of alphabetic words, which ends with the word STOP. STOP is not part of the initial sentence nor will it occur as part of any option later. Then you will have a menu in a loop with the following options (taken as string inputs):

- EXIT: This will exit from the menu loop and the program
- INSB: This will take as input a string st1 and insert the string st1 at the beginning of the sentence, making it the first word of the sentence.
- INSE: This will take as input a string st1 and insert the string st1 at the end of the sentence, making it the last word of the sentence.
- DEL: This will take as input a string st1 and delete the first occurrence of a word in the sentence that has the string st1 as a substring.
- REP: This will take as input two strings st1 and st2 and replace the last occurrence of the word in the sentence which has st1 as a substring by the word st2. The whole word will be replaced by st2 if the substring match condition is satisfied.
- DISP: This will print the sentence neatly, assuming that a line has no more than 50 characters, preferably with left and right adjusted alignment characters, creating suitable gaps in between to ensure that the display appears pretty.
- UNDO: It will undo the last operator, by choosing the previous option used other than DISP. Between two UNDO options, there must be some option other than DISP which is to be undone. Nor can UNDO be the first option after reading the initial sentence.

Save the file as A15_<Roll Number>.c (example A15_21AG10002.c). Build, Run and Test it on your own data. Then upload the .c file for the Assignment.

[40 Marks: 15 marks for intermediate submission, 25 marks for final submission]

In particular, you will do the following: [Assume a word can have at most 10 characters] Initial sentence is input only once

1. Define the data organization for storing the sentences and other information in the most optimal manner possible. There is no assumption on the maximum number of words. However, if you are unable to design it without a maximum limit on the number of words, then you may read in an integer n and assume that there will never be more than n words. Use dynamic allocation.
2. Write functions for each of the above options. If you need to write some other functions, you may do so.
3. In the main program, read in the initial sentence and write a menu structure in a suitable loop to take the options as string inputs and call the appropriate functions. Check for wrong options and ask for option again.

[Do not call any Library Functions other than scanf, printf. You may use your own user defined functions.]

Test Data: Test it with your own data for various cases.