The assignment will be graded out of 100 points.

Due: 11:59pm, Wednesday, June 27

Submission Guidelines:

Assignment will be submitted via Blackboard. If there are multiple files that you are submitting then you should zip your files into a single file and submit it. The name of this file should be in following format: lastname_firstname_UTANetID. If you are submitting a single file, make sure it is in the following format: word, txt or pdf and it should be named as mentioned above.

Make sure your name and your student ID are listed in your assignments.

Deductions for failing to follow directions:

If your assignment is not completed by the deadline, send it anyway and review it with the TA for partial credit. Do not take a zero or excessive late penalties just because it isn't working yet. We will make an effort to grade you on the work you have done.

Note: Please do not forget to zip all the files in ONE folder

Example:

Task1.c
Task2.c
Task3.c
Task4.c
Task5.c
Task6.c

After that, copy all the C files into one folder and compress(zip) the folder. Then, submit the compressed (zipped) folder on Blackboard.

Assignment Specification:

Task 1 (20 pts.)

In a file called task1.c, write a C program regarding the following requirements:

You are given an array of integers “array []” and an input “x”. You have to find if there are any two numbers in your array such that the sum of their square equals to x^2. In other words, you have to find out if there is any pair of i,j such that $array[i]^2 + array[j]^2 = x^2$. If there are such pairs, you should print all such (i,j). Otherwise print “There are no such pairs”.

Example: array []: 6, -4, 6, 3, 9, 0, -1, -9, 2, 6

x: 5

You have 1 pair: $(-4)^2 + (3)^2 = 5^2$

So, your output would be corresponding array indices: (1, 3)

Note: The array size is 10 and you have to get the elements of the array from user. You also need to take the value of “x” from user.

Task 2 (20 pts.)

In a file called task2.c, write a C program to generate Fibonacci numbers as many as desired. After the construction and storing them in an array, you can print the sum value of the sequence.

The example is below and user input is in **boldface**.

Enter the number of terms: **9**
First 7 terms of Fibonacci numbers are: 0 1 1 2 3 5 8 13 21
The sum value of the above sequence is: 54

Note: **Fibonacci sequence** has many applications in computer science. First few numbers are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, etc., except first two terms in sequence

Task 3 (20 pts.)

In a file called task3.c, write a C Program to calculate the sum & difference of the 2-D matrices. The program should first read 2 matrices and then performs both addition and subtraction of matrices.

Example:
Enter number of row: 3
Enter number of column: 2

Enter value for matrix 1:
8
7
3
2
0
6

Enter value for matrix 2:
2
4
9
6
3
8

Sum of 2 matrices is:
10 11
12 8
3 14

Difference of 2 matrices is:
6 3
-6 -4
-3 -2

Task 4 (10 pts.)

In a file called task4.c, write a C program to get 10 decimal integer values (both positive and negative) from user, and store them in an array. Then, use “bubble sort” to sort out the array and print the numbers on the screen.

Example:
Input Numbers: 1 2 -19 5 -7 11 22 -13 15 28

Expected output:
Sorted List: -19, -13, -7, 1, 2, 5, 11, 15, 22, 28

Task 5 (20 pts.)

You are given a list (array) of n-1 integers and these integers are in the range of 1 to n. There are no duplicates in list. One of the integers is missing in the array. In a file called task5.c, write a C program to find the missing integer.

The example is below and user input is in **boldface**.

Example:

Enter number of elements in array(n): **6**
Input (n-1):
1
2
3
5
6

Expected output: 4

Hint:
1. Get the sum of numbers: SUM = n*(n+1)/2
2. Subtract all the numbers from sum and you will get the missing number.

Task 6 (10 pts.)

You are given an array of 0s and 1s in random order. In a file called task6.c, write a C program to segregate 0s on left side and 1s on right side of the array (Traverse array only once).

Example:

Input array = [0, 1, 0, 1, 0, 0, 1, 0, 1, 0]

Expected output:
Output array= [0, 0, 0, 0, 0, 0, 1, 1, 1, 1]

Assignment Guidelines:

There will be several programming assignments in this course, typically assigned on a weekly basis. All assignments will have equal weight. No assignment scores will be dropped. The following class policies regarding assignments will be followed:

- All assignments must be submitted via Blackboard.
- No deadline extensions for the entire class will be provided. (See syllabus about policy on extensions for individuals, based on emergencies documented in writing).
- No extra credit will be provided.
- If you make multiple submissions to Blackboard for the same assignment, only the latest submission will be graded.

Late submission policy:

- All assignments are graded out of 100 points. Assignments submitted late will be penalized, at a rate of 4 penalty points per hour. The submission time will be the time shown on Blackboard. Any assignment submitted more than 25 hours late will receive no credit.
- Exceptions to late submission penalties will only be made for emergencies documented in writing, in strict adherence to UTA policy. For all such exception requests, the student must demonstrate that he or she made all efforts to notify the instructor as early as possible.
- Computer crashes, network crashes, and software or hardware failure will NOT be accepted as justification for late submissions. If you want to minimize chances of a late submission, aim to submit early. You can always revise your submission till the deadline.
- Sometimes students submit the wrong files on Blackboard. Unfortunately, no credit or waiver of late penalties can be provided in such cases.
- If you find yourself in an emergency situation and cannot deliver homework on time, immediately inform the instructor and teaching assistant. Even if you have a valid reason for delivering late an assignment, you must make a convincing case that you have notified the instructor and teaching assistant as early as possible.

If you want to minimize chances of a late submission, aim to submit early. You can always revise your submission till the deadline.