**Rules**

1. Submission Steps:
   1. In your local machine, create a new folder; the name of the folder should be your 7 digit roll number.
   2. For each problem that you choose to solve, code your solution in a .c file. The name of the file should be exactly as is mentioned in problem description. Put this C file in the folder created in step 1(a).
   3. Finally, zip the folder created in step 1(a) to produce a .zip file. The name of the .zip file should be your 7 digit roll number.
   4. Submit the .zip file created in step 1(c).
2. You must submit your solution package (.zip file) by 10:00pm 12/07/2019 through Moodle.
3. You must bring your solution package in the next lab class. sYou must be able to explain your code.
4. Do not copy code. You will be caught and given -20.

**Problem #1**

**Mode**

**File name: mode.c**

Find the mode of some given numbers. The mode of some numbers is/are the ones which appear the highest number of times. Take an integer *n* as input. Next take *n* integer elements as input, whose mode you need to find. All the inputs will have values in [0, 10000] range. (Hint: Note the upper limit; you can count frequencies for this.) Print the mode(s) of the given numbers. **Use pointer arithmetic and dynamic memory allocation. Do not use array of some predefined size. Do not use array indexing.**

|  |  |
| --- | --- |
| **Sample Input(s)** | **Corresponding Output(s)** |
| 3  1 3 1 | 1 |
| 4  2 2 1 4 | 2 |
| 5  1 2 1 4 4 | 1 4 |

**Problem #2**

**Odd even sum**

***File name: sum.c***

Given some integers, find if the sum of the integers at odd positions is greater than or less than or equal to that of those at even positions. **You must use pointers to solve this problem. Do not use array indexing**. (Assume indexing starts from 0). Take an integer *n* as input. Next line contains *n* integer which are the integers to be considered.

|  |  |
| --- | --- |
| **Sample Input(s)** | **Corresponding Output(s)** |
| 3  1 3 1 | Odd index sum is greater |
| 4  2 2 4 1 | Even index sum is greater |
| 5  1 2 1 4 4 | Equal |

**Problem #3**

**Encryption**

***File name: encryotion.c***

Encryption is a mechanism of hiding a piece of information in such a way that only who knows how to discover its hidden meaning can extract the information. Suppose you want to send an encrypted message. As you know about pointers, you are going to use them for your encryption.

You already know that a character takes 1 byte while a long long int takes 8 bytes of memory. (The format specifier for long long int is “%lld”). Suppose you have a text message stored in a character array. If you consider 8 consecutive bytes of that array together you can get a long long integer value based on the ASCII value of the characters and their positions. So if your message consists of 16 characters, you can encrypt them in two long long integers and send it as the encrypted text for passing hidden information!!

**Input Specification**

Take the length of your message, *n* as input followed by your message with *n* characters. You can safely assume that *n* will always be given as a multiple of 8.

**Sample Input-Output**

|  |  |
| --- | --- |
| **Sample Input(s)** | **Corresponding Output(s)** |
| 16  Hello Programmer | 8237119381393139016 8243115009525311343 |
| 24  I hate this problem! :'( | 8367799658179207241 7093013764038158696 2893345195283015020 |