**Assignment no.3**

**Submission date: Thu 22/08/2024**

**Q1:** Write a function called **picker** that takes three input arguments called **condition**, **in1** and **in2** in this order. The argument **condition** is a logical. If it is true, the function assigns the value of **in1** to the output argument**out,**otherwise, it assigns the value of **in2** to**out**. See the examples below to see how **picker** works in practice.

**Q2:** Write a function called **eligible** that helps the admission officer of the Graduate School of Vanderbilt University decide whether the applicant is eligible for admission based on GRE scores. The function takes two positive scalars called **v** and **q** as input and returns the logical **admit** as output. They represent the percentiles of the verbal and quantitative portions of the GRE respectively. You do not need to check the inputs. The applicant is eligible if the average percentile is at least 92% and both of the individual percentiles are over 88%. The function returns logical true or false value.

**Q3:** Write a function called **valid\_date** that takes three positive integer scalar inputs **year**, **month**, **day**. If these three represent a valid date, return a logical **true**, otherwise **false**. The name of the output argument is **valid**. If any of the inputs is not a positive integer scalar, return **false** as well. Note that every year that is exactly divisible by 4 is a leap year, except for years that are exactly divisible by 100. However, years that are exactly divisible by 400 are also leap years. For example, the year 1900 was not leap year, but the year 2000 was. Note that your solution must not contain any of the date related built-in MATLAB functions.

**Q4:** Write a function called **next\_prime** that takes a scalar positive integer input **n**. Use a *while-loop* to find and return **k**, the smallest prime number that is greater than **n**. Feel free to use the built-in **isprime** function. Here are some example runs:

**Q5:** Write a function called **max\_sum** that takes **v**, a row vector of numbers, and **n**, a positive integer as inputs. The function needs to find the **n** consecutive elements of **v** whose sum is the largest possible. In other words, if **v** is **[1 2 3 4 5 4 3 2 1]** and **n** is 3, it will find 4 5 and 4 because their sum of 13 is the largest of any 3 consecutive elements of **v**. If multiple such sequences exist in **v**, **max\_sum** returns the first one. The function returns **summa**, the sum as the first output argument and **index**, the index of the first element of the **n** consecutive ones as the second output. If the input **n** is larger than the number of elements of **v**, the function returns 0 as the sum and -1 as the index. Here are a few example runs:

**Note:** Please submit this assignment as a PDF, including both your code and the corresponding results. Kindly share the file via WhatsApp at 03096078248.