1. (20 points) Define and initialize a variable of type integer. Then, compute and print the sum of first and last digit. For example, if the number is 5492, the sum would be 2+5=7. You are not allowed to change the data type of the variable or assign it to another variable of another data type. Note that the solutions should work for any integer number, i.e. is not dependent on the number of digits.

Note: if you prefer to receive the integer value from input, you can use scanf() function like this

scanf(“enter a number %d\n”, variable\_name);

2. (20 points) Write a function that receives a string and return a double floating-point value. The function declaration would look like this

double convertFloat(char s[]);

Do not use the strtol() function or any other standard C library function. Write your own! Include a test program for the function. Note that the input string in

3. (25 points) Using the following algorithm, write a C program that prints all prime numbers less than n.

* Create an array of integers from 2 to n.
* Start with 2, the first number in the array (and the smallest prime). Keeping 2 in the list, knock out all multiples of 2, as these can't be prime.
* The next number in the array is 3. Again keeping 3 in the array, knock out all multiples of 3.
* 4 has been knocked out (as a multiple of 2), so the next number in the list is 5. Keep 5 and remove its multiples.
* And so on. At the end, only the primes less than or equal to n are left.

4. (10 points) For all ASCII characters with decimal values between 33 to 126, write a C program that prints the decimal value and the corresponding character.

5. (25 points) The **Euclidean distance** between points **p** and **q** is the length of the line segment connecting them in Cartesian system.

If **p** = (*p*1, *p*2,..., *pn*) and **q** = (*q*1, *q*2,..., *qn*) are two points in Euclidean *n*-space, then the distance (d) from **p** to **q**, or from **q** to **p,** in general, for an *n*-dimensional space, is

d(p, q) = \sqrt{(p_1- q_1)^2 + (p_2 - q_2)^2+\cdots+(p_i - q_i)^2+\cdots+(p_n - q_n)^2}.

Write a C program to calculate the **Euclidean distance** for two points in 3-dimensional space.

### Assign your own values as the coordinates of the points p and q for test examples.

### You are **NOT** allowed to use sqrt() and pow() functions of C math library. Develop your own functions for those. For square root, use Newton’s method. You can find more about the method here. <https://en.wikipedia.org/wiki/Newton%27s_method#Square_root_of_a_number>

3-digit accuracy after the decimal point in all floating-point operations is enough.