CS348-Assignment 1

Name: Sweeya Reddy

Roll no. : 200101079

README

Submission Contents:

- **A1A_200101079.asm**
- **hello.txt**
- **A1B_200101079.asm**
- **Readme**

Running the code:

- **A1A_200101079.asm**:

  nasm -felf64 A1A_200101079.asm && gcc -no-pie A1A_200101079.o && ./a.out

```
sweeya@sweeya-Inspiron-5590:~/Desktop$ nasm -felf64 A1A_200101079.asm && gcc -no-pie A1A_200101079.o && ./a.out
```

- **A1B_200101079.asm**:

  nasm -felf64 A1B_200101079.asm && gcc -no-pie A1B_200101079.o && ./a.out

```
sweeya@sweeya-Inspiron-5590:~/Desktop$ nasm -felf64 A1B_200101079.asm && gcc -no-pie A1B_200101079.o && ./a.out
```

System and Version Requirements:

Linux (Ubuntu 20.04 LTS), NASM version 2.14.02

Note: Gcc is needed to use scanf and printf appropriately which are declared under the .text with keyword extern.

Question 1

- All inputs are assumed to be integral in the 4 byte range.
- The entire code can be divided into the following steps:
    - OPENING AND READING THE FILE AND STORING IT IN BUFFER
      Note: The File Is Hello.Txt In This Case
    - ITERATING OVER EACH CHARACTER OF BUFFER
    - CHECKING IF THE CURRENT CHARACTER IS A ALPHABET
    - CHECKING IF THE CURRENT CHARACTER IS A NUMBER
    - IF THE CHARACTER IS NEITHER A ALPHABET OR A NUMBER
    - PRINTING THE NUMBER OF ALPHABETS AND THE ALL THE ALPHABETS IN THE FILE
    - PRINTING THE NUMBER OF NUMBERS AND THE ALL THE NUMBERS IN THE FILE
    - PRINTING THE NUMBER OF SPECIAL SYMBOLS AND THE ALL THE SYMBOLS IN THE FILE
    - CLOSING THE FILE

      Note: The steps are shown in the comments of the code in uppercase.
- In the section .data the variable file has the name of the file from which we are trying to read.

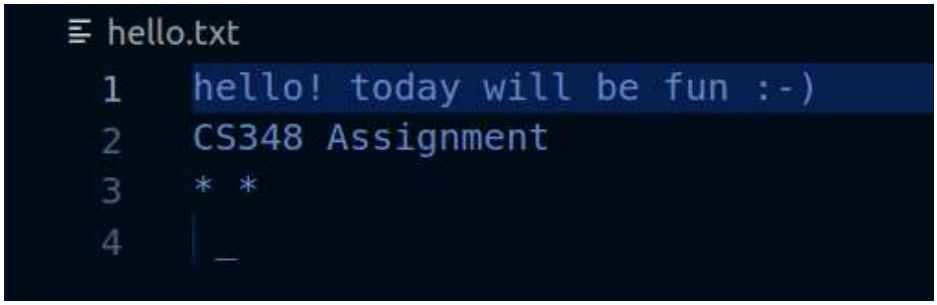
  Note: Make sure the .asm file and the .txt file are from the same directory.
- Then once we run the command on the terminal, we get our output.
- The output consists of:
    - 1st line: The number of alphabets
    - 2nd line: All the alphabets in the file
    - 3rd line: The number of numbers
    - 4th line: All the numbers in the file
    - 5th line: The number of special symbols
    - 6th line: All the special symbols in the file
- Note: While reading the file, the length of the file must be mentioned. As the file size can vary, 2024 bytes has been set as the size. This can be the maximum length of the buffer.

SCREENSHOT OF OUTPUT UNDER DIFFERENT TEST CASES:

Example 1:


```
sweeya@sweeya-Inspiron-5590:~/Desktop$ nasm -felf64 A1A_200101079.asm && gcc -no-pie A1A_200101079.o && ./a.out
The number of alphabets are: 9
h e l l o D a t e
The number of numbers are: 5
2 3 1 2 3
The number of special characters are: 9
!   : )
 :   : :
```

Example 2:



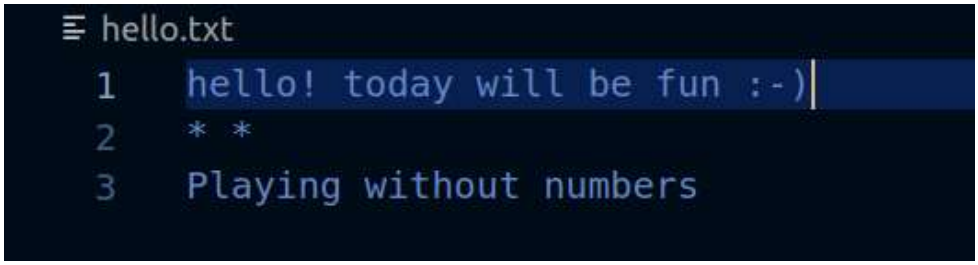

```
sweeya@sweeya-Inspiron-5590:~/Desktop$ nasm -felf64 A1A_200101079.asm && gcc -no-pie A1A_200101079.o && ./a.out
The number of alphabets are: 31
h e l l o t o d a y w i l l b e f u n C S A s s i g n m e n t
The number of numbers are: 3
3 4 8
The number of special characters are: 18
!          : - )

 *   *
```

Corner cases:

Example 3:

When one of the 3 counts is 0.

Here, I have no numbers in my hello.txt file.




```
sweeya@sweeya-Inspiron-5590:~/Desktop$ nasm -felf64 A1A_200101079.asm && gcc -no-pie A1A_200101079.o && ./a.out
The number of alphabets are: 40
h e l l o t o d a y w i l l b e f u n P l a y i n g w i t h o u t n u m b e r s
The number of numbers are: 0

The number of special characters are: 16
!          : - )
 *   *
```

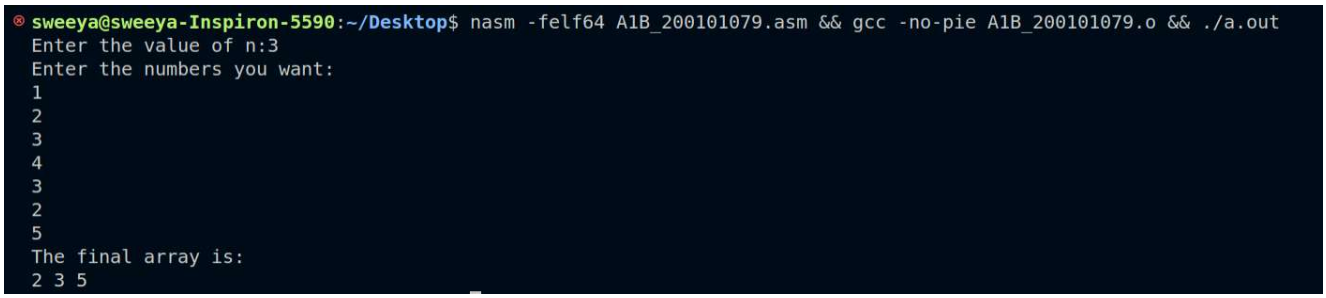
Question 2

- All inputs are assumed to be integral in the 4 byte range.
- The entire code can be divided into the following steps:
    - TAKE INPUT N
    - TAKE INPUT NUMBERS IN A LOOP
    - CHECK CORNER CASE
    - CHECK IF THE GIVEN NUMBER IS A PRIME
    - CHECK IF THE GIVEN NUMBER IS A DUPLICATE
    - APPENDING NUMBER INTO ARRAY IF THE TWO CONDITIONS ARE SATISFIED
    - OUTPUTTING THE FINAL ARRAY

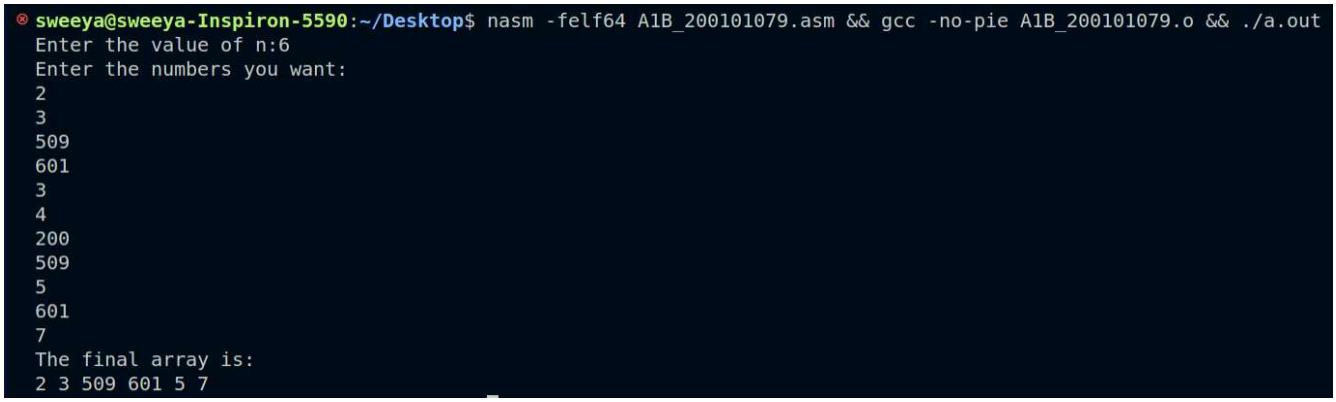
      Note: The steps are shown in the comments of the code in uppercase.
- On running the command, we get a prompt on the terminal to enter the value of n.
- Then you get a prompt to enter all the numbers.
- Then the corner case is checked, i.e. if the number is less than or equal to 1, then discard it and go to the next number.
- Primality testing is done if any number from 2 to that number-1, divides the number. If it does then the number is not prime and we go to the next number.
- To check if the number is a duplicate, the number is compared with all the numbers which are present in the input_array. If found equal, then we go to the next number.
- Each number is 4 bytes, and it is appended at the end of the input_array (whose address is input_array + 4*eax, where eax stores the value of the counter which is the number of numbers already present in the input_array) if the two conditions are satisfied.
- Once there are n numbers in the array which satisfy primality and non duplicates, then no more input numbers are taken.
- Then the final array is outputted.

SCREENSHOT OF OUTPUT UNDER DIFFERENT TEST CASES:

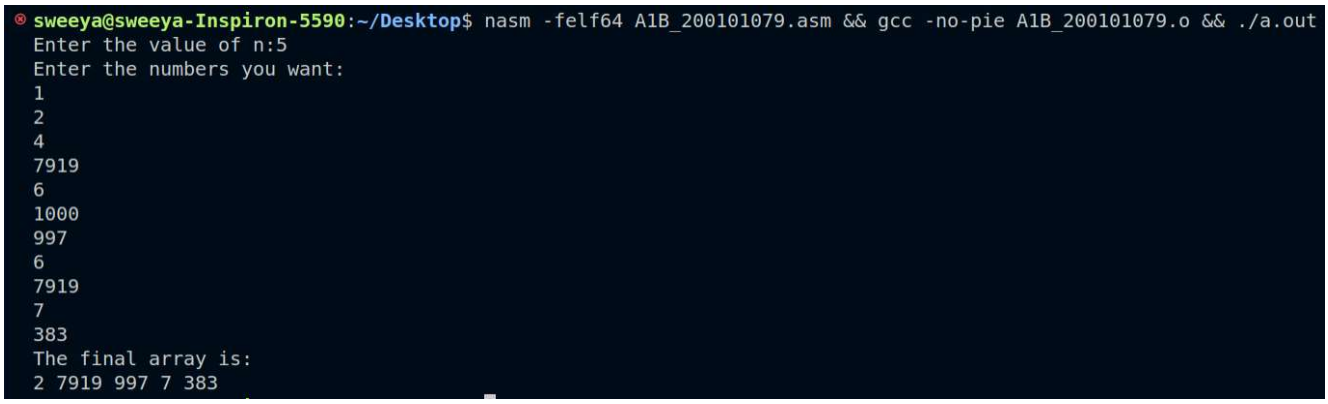
Example 1:



Example 2:



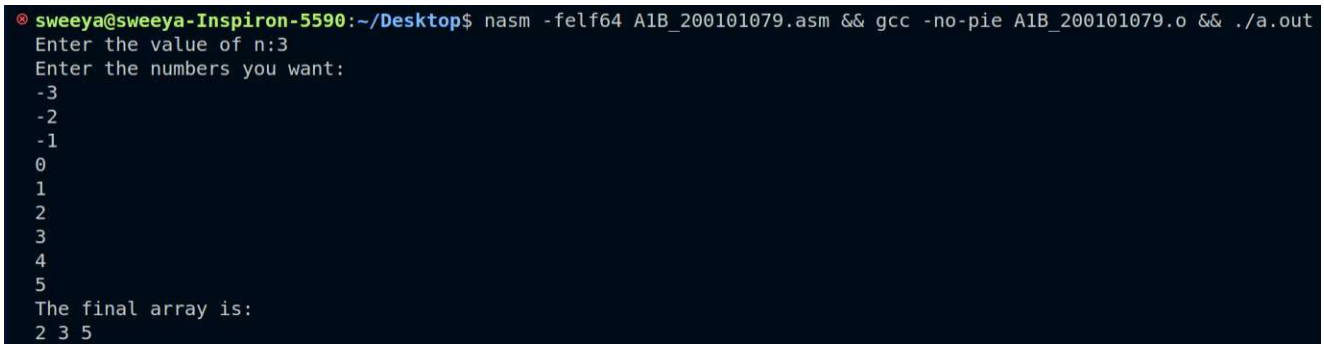
Example 3:



Corner case:

Example 4:

When negative numbers are present, we do not consider that number.



THANK YOU