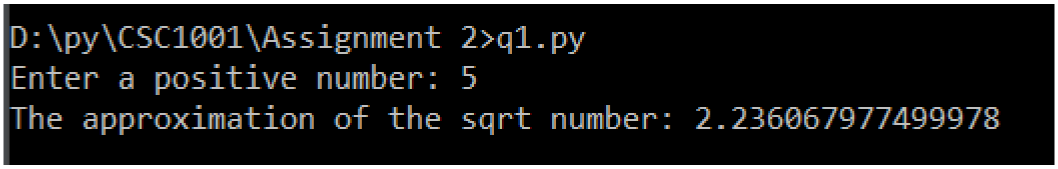
Question 1:

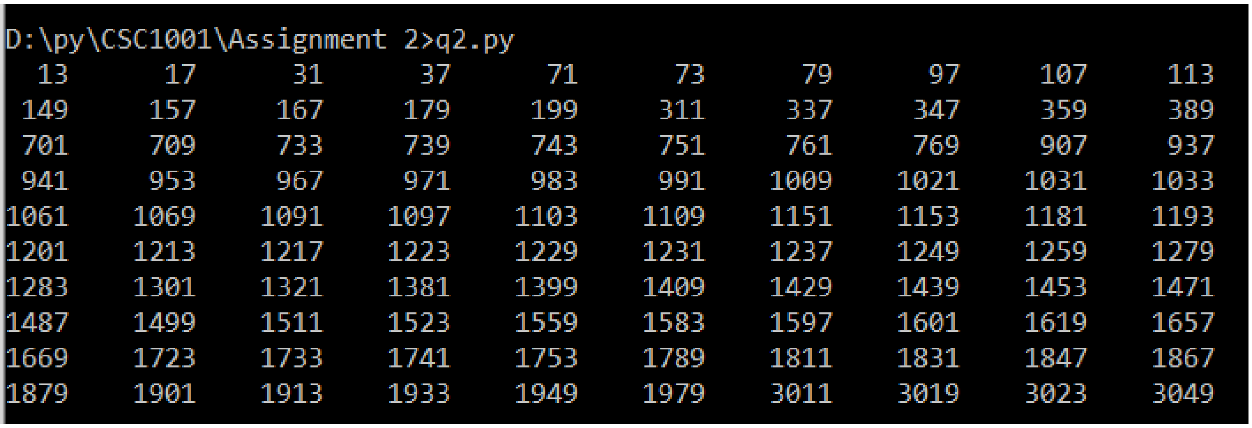
1. This code is saved in *q1.py*
2. - This program allows user to input a positive number n.
   - This program prompts the user to input the value as a positive number.
   - If the user fails to input the value in integer / float, an error message "*The input should be a number*" will show up and the program will ask the user to reinput the value.
   - If the integer value is not a positive number, an error message "*The number should be positive*" will show up and the program will ask the user to reinput the value.
   - This program outputs the approximation of square root of number using the Babylonian function.
3. Execute as followings:



```
D:\py\CSC1001\Assignment 2>q1.py
Enter a positive number: 5
The approximation of the sqrt number: 2.23606797749978
```

Question 2:

1. This code is saved in *q2.py*
2. - This program does not allow any inputs from the user.
   - This program outputs 100 *emirp* numbers (*emirp* is defined as a nonpalindromic prime number whose reversal is also a prime).
3. Execute as followings:



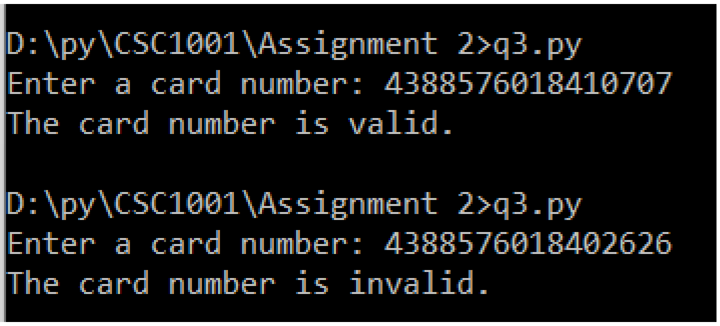
```
D:\py\CSC1001\Assignment 2>q2.py
  13      17      31      37      71      73      79      97     107     113
 149     157     167     179     199     311     337     347     359     389
 701     709     733     739     743     751     761     769     907     937
 941     953     967     971     983     991    1009    1021    1031    1033
1061    1069    1091    1097    1103    1109    1151    1153    1181    1193
1201    1213    1217    1223    1229    1231    1237    1249    1259    1279
1283    1301    1321    1381    1399    1409    1429    1439    1453    1471
1487    1499    1511    1523    1559    1583    1597    1601    1619    1657
1669    1723    1733    1741    1753    1789    1811    1831    1847    1867
1879    1901    1913    1933    1949    1979    3011    3019    3023    3049
```

Question 3:

1. This code is saved in *q3.py*
2. - This program allows user to input a number.
   - This program prompts the user to input the value as a positive number with length between 13 and 16 digits.
   - If the user fails to input a positive number, an error message "*The input should be a positive integer number*" will show up and the program will ask the user to reinput the value.

- If the user fails to input the positive number with length between 13 and 16 digits, an error message "*The length number should be between 13 and 16*" will show up and the program will ask the user to reinput the value.
- This program outputs whether the card number is valid or not using algorithm that was proposed by Hans Luhn of IBM in 1954.

3. Execute as followings:

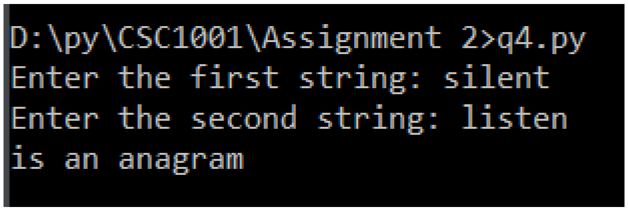


```
D:\py\CSC1001\Assignment 2>q3.py
Enter a card number: 4388576018410707
The card number is valid.

D:\py\CSC1001\Assignment 2>q3.py
Enter a card number: 4388576018402626
The card number is invalid.
```

Question 4:

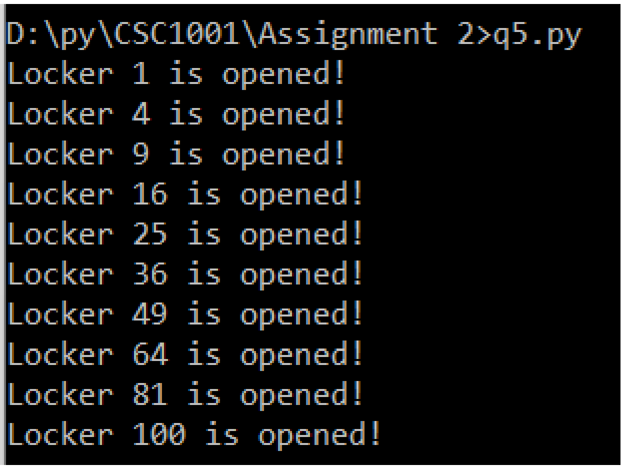
1. This code is saved in *q4.py*
2. - This program allows user to input two strings.
   - This program prompts the user to input both strings containing only letters.
   - If the user fails to input a string that contains only letters, an error message "*The string may only consists of alphabet characters*" will show up and the program will ask the user to reinput the string.
   - This program outputs whether both strings are anagrams or not.
3. Execute as followings:



```
D:\py\CSC1001\Assignment 2>q4.py
Enter the first string: silent
Enter the second string: listen
is an anagram
```

Question 5:

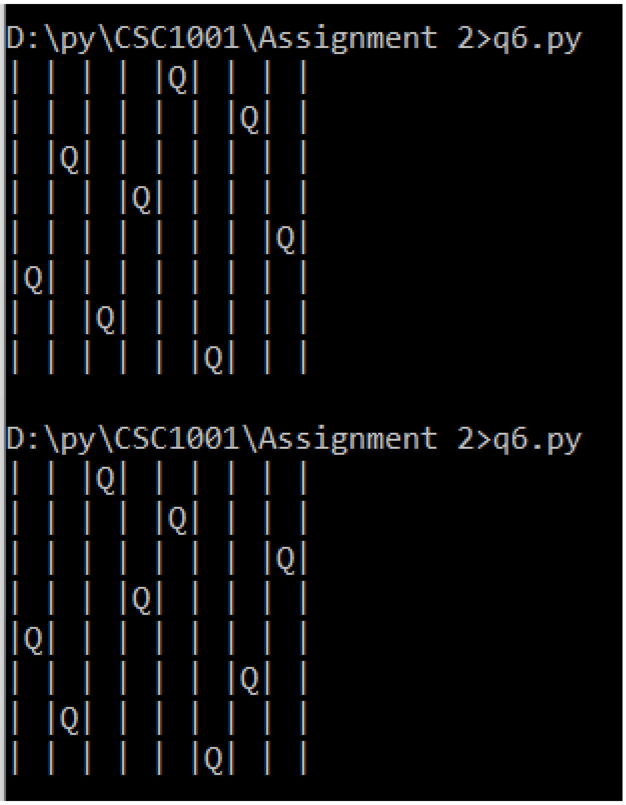
1. This code is saved in *q5.py*
2. - This program does not allow any inputs from the user.
   - This program outputs lockers that are opened after the 100th operation is done. The ith operation toggle all lockers starting from i, 2i, 3i, …, ki (ki <= 100).
3. Execute as followings:



```
D:\py\CSC1001\Assignment 2>q5.py
Locker 1 is opened!
Locker 4 is opened!
Locker 9 is opened!
Locker 16 is opened!
Locker 25 is opened!
Locker 36 is opened!
Locker 49 is opened!
Locker 64 is opened!
Locker 81 is opened!
Locker 100 is opened!
```

Question 6:

1. This code is saved in *q6.py*
2. - This program does not allow any inputs from the user.
   - This program outputs one of the solutions for the classic eight queen puzzle (placing eight queens on a chessboard (8*8) such that no two queens can attack each other).
   - This program may outputs a different solution as the random algorithm is used inside the code.
3. Execute as followings:



```
D:\py\CSC1001\Assignment 2>q6.py
| | | | |Q| | | |
| | | | | | |Q| |
| |Q| | | | | | |
| | | |Q| | | | |
| | | | | | | |Q|
|Q| | | | | | | |
| | |Q| | | | | |
| | | | | |Q| | |

D:\py\CSC1001\Assignment 2>q6.py
| | |Q| | | | | |
| | | | |Q| | | |
| | | | | | | |Q|
| | | |Q| | | | |
|Q| | | | | | | |
| | | | | | |Q| |
| |Q| | | | | | |
| | | | | |Q| | |
```