1. Write a program to find the values that form a traingle.

Input: Three lists containing the lengths of sides of traingle.

Output: set((x,y,z)/x,y,z forms a traingle and x,y,z belongs to list1,list2,list3 respectively)

2. Finding the prime numbers between n,m

Input: Two integers n,m

Output: Prime numbers between n,m

3. Write a program to find whether the multiplication of two numbers in the given list is also present in the list

Input: A list of N numbers is given

Output: set((x,y)/x,y,x*y belongs to list)

4. Program should take the input as string and print the result in the below form.

Input: str="asdfghjkl"

Output: ['a', 'as', 's', 'asd', 'sd', 'd', 'asdf', 'sdf', 'df', 'f', 'asdfg', 'sdfg', 'dfg', 'fg', 'g', 'asdfgh', 'sdfgh', 'dfgh', 'fgh', 'gh', 'h', 'asdfghj', 'sdfghj', 'dfghj', 'fghj', 'ghj', 'hj', 'j', 'asdfghjk', 'sdfghjk', 'dfghjk', 'fghjk', 'ghjk', 'hjk', 'jk', 'k', 'asdfghjkl', 'sdfghjkl', 'dfghjkl', 'fghjkl', 'ghjkl', 'hjkl', 'jkl', 'kl', 'l']

5. Given a list of N numbers, use a single list comprehension to produce a new list that only contains those values that are:

(a) even numbers, and

(b) from elements in the original list that had even indices

For example, if list[2] contains a value that is even, that value should be included in the new list, since it is also at an even index (i.e., 2) in the original list. However, if list[3] contains an even number, that number should not be included in the new list since it is at an odd index (i.e., 3) in the original list.

6. What is the output of the following?

```
k = [print(i) for i in my_string if i not in "aeiou"]
```

7. What is the output of print(k) in the following?

```
k = [print(i) for i in my_string if i not in "aeiou"]
print(k)
```

8. What is the output of the following?

```
my_string = "hello world"
k = [(i.upper(), len(i)) for i in my_string]
print(k)
```

9. What is the output of the following?

```
x = [i**+1 for i in range(3)]; print(x);
```

10. What is the output of the following?

```
print([i.lower() for i in "HELLO"])
```

11. What is the output of the following?

```
print([i+j for i in "abc" for j in "def"])
```

12. What is the output of the following?

```
print([[i+j for i in "abc"] for j in "def"])
```

13. What is the output of the following?

```
print([if i%2==0: i; else: i+1; for i in range(4)])
```

14. What is the output of the following?

```
list(map(lambda x: x**-1, [1, 2, 3]))
```

Write equivalent sentence using list comprehensions (i.e. without using map and lambda).

15. Compute followings using list comprehensions:

(a) Sum of the first n *counting* numbers: $1+2+3+ \ldots +n$

(b) Sum of the first n odd numbers: $1+3+5+ \ldots +(2n-1)$

(c) Sum of a series of numbers entered by the user until the value 999 is entered.

Note: 999 should not be part of the sum.

(d) The number of times a whole number n can be divided by 2 (using integer division) before reaching 1.

16. Implement bubble sort using list.

17. Write a program to determine how long it takes for an investment to double at a given interest rate. The input will be an annualized interest rate, and the output is the number of years it takes an investment to double. Note: the amount of the initial investment does not matter.

18. Heating and cooling degree-days are measures used by utility companies to estimate energy requirements. If the average temperature for a day is below 60, then the number of degrees below 60 is added to the heating degree-days. If the temperature is above 80, the amount over 80 is added to the cooling degree-days. Write a program that accepts a sequence of average daily temps and computes the running total of cooling and heating degree-days. The program should print these two totals after all the data has been processed.

19. The math library contains a function that computes the square root of numbers. Here, you are to write your own algorithm for computing square roots. One way to solve this problem is to use a guess-and-check approach. You first guess what the square root might be and then see how close your guess is. You can use this information to make another guess and continue guessing until you have found the square root (or a close approximation to it). One particularly good way of making guesses is to use Newton's method. Suppose x is the number we want the root of, and guess is the current guessed answer. The guess can be improved by using $(guess + x/guess)/2$ as the next guess. Write a program that implements Newton's method. The program should prompt the user for the value to find the square root of (x) and the number of times to improve the guess. Starting with a guess value of $x/2$, your program should loop the specified number of times applying Newton's method and report the final value of guess.