C++ And Python Test-question And Answers

1.A library maintains a record of reader ratings for its collection of books. The ratings are stored in an array, where each element represents the rating of a book on a scale of 1 to 5.

Code

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main() {
    int M;
    cout << "Enter the number of books: ";
    cin >> M;
    if (M < 1 || M > 1000) {
        cout << "Invalid number of books. Must be between 1 and 1000." << endl;
        return 1;
    }
    vector<int> ratings(M);
    cout << "Enter the ratings (1 to 5): ";
    for (int i = 0; i < M; ++i) {
        cin >> ratings[i];
        if (ratings[i] < 1 || ratings[i] > 5) {
            cout << "Invalid rating. Must be between 1 and 5." << endl;
            return 1;
        }
    }
    double sum = 0;
    for (int r : ratings) {
        sum += r;
    }
```

```
    double average = sum / M;
    cout << "Average rating: " << average << endl;
    int goodCount = 0;
    for (int r : ratings) {
        if (r >= 4) ++goodCount;
    }
    cout << "Number of books with good ratings (4 or 5): " << goodCount << endl;
    int highest = ratings[0], lowest = ratings[0];
    int highIndex = 0, lowIndex = 0;
    for (int i = 1; i < M; ++i) {
        if (ratings[i] > highest) {
            highest = ratings[i];
            highIndex = i;
        }
        if (ratings[i] < lowest) {
            lowest = ratings[i];
            lowIndex = i;
        }
    }
    cout << "Highest rating: " << highest << " at index " << highIndex << endl;
    cout << "Lowest rating: " << lowest << " at index " << lowIndex << endl;
    sort(ratings.begin(), ratings.end(), greater<int>());
    cout << "Ratings in descending order: ";
    for (int r : ratings) {
        cout << r << " ";
    }
    cout << endl;

    return 0;
}
```

Output

Enter the number of books: 5

Enter the ratings (1 to 5): 5

4

3

5

4

Average rating: 4.2

Number of books with good ratings (4 or 5): 4

Highest rating: 5 at index 0

Lowest rating: 3 at index 2

Ratings in descending order: 5 5 4 4 3

2.You have a **Juice Bottle** that initially contains **500 ml** of juice. You want to simulate refilling this bottle using a class-based approach.

Create a class named RefillJuice with a data member named volume initialized to **500 ml**.

Code

```
#include <iostream>

using namespace std;

class RefillJuice {

private:

   int volume;

public:

   RefillJuice() {

      volume = 500;

      cout << "Final volume: " << volume << " ml" << endl;

   }

   RefillJuice(int addedJuice) {

      volume = 500 + addedJuice;
```

```
        cout << "Final volume: " << volume << " ml" << endl;
    }
};
int main() {
    RefillJuice bottle1;
    RefillJuice bottle2(250);
    return 0;
}
```

Output

Final volume: 500 ml

Final volume: 750 ml

3.A university wants to manage its academic staff using an object-oriented approach in C++. The university has a hierarchical structure where a Person can be a general entity with basic details, a Teacher inherits from Person to add teaching-related information, and a Professor further inherits from Teacher to include additional attributes related to research. To implement this, create a base class Person with attributes name (a string) and age (an integer), along with a constructor to initialize these attributes and a display() function to print personal details. Next, create a derived class Teacher that inherits from Person and adds an attribute subject (a string) to represent the subject they teach. This class should also override the display() function to print both personal details and subject information. Finally, create another derived class Professor, which inherits from Teacher, and adds an attribute researchArea (a string) to store the professor's research specialization. The display() function in Professor should be overridden to print all details, including the research area. In the main() function, create an object of the Professor class, initialize it with appropriate values, and call the display() function to print all details.

Code

```
#include <iostream>
#include <string>
using namespace std;
class Person {
protected:
    string name;
    int age;
```

```
public:
    Person(string n, int a) : name(n), age(a) {}
    virtual void display() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};
class Teacher : public Person {
protected:
    string subject;
public:
    Teacher(string n, int a, string s) : Person(n, a), subject(s) {}
    void display() override {
        Person::display();
        cout << "Subject: " << subject << endl;
    }
};
class Professor : public Teacher {
private:
    string researchArea;
public:
    Professor(string n, int a, string s, string r) : Teacher(n, a, s), researchArea(r) {}
    void display() override {
        Teacher::display();
        cout << "Research Area: " << researchArea << endl;
    }
};
int main() {
    Professor prof("Dr. Ananya Sharma", 45, "Computer Science", "Artificial Intelligence");
    prof.display();
    return 0;
```

```
}
```

Output

```
Name: Chethan Kumar
Age: 23
Subject: Computer Science
Research Area: Artificial Intelligence
```

1.Write a program which will find all such numbers which are divisible by 7 but are not a multiple of 5, between 2000 and 3200 (both included). The numbers obtained should be printed in a comma-separated sequence on a single line.

```
numbers = [str(i) for i in range(2000, 3201) if i % 7 == 0 and i % 5 !
= 0]
print(",".join(numbers))
```

```
2002,2009,2016,2023,2037,2044,2051,2058,2072,2079,2086,2093,2107,2114,
2121,2128,2142,2149,2156,2163,2177,2184,2191,2198,2212,2219,2226,2233,
2247,2254,2261,2268,2282,2289,2296,2303,2317,2324,2331,2338,2352,2359,
2366,2373,2387,2394,2401,2408,2422,2429,2436,2443,2457,2464,2471,2478,
2492,2499,2506,2513,2527,2534,2541,2548,2562,2569,2576,2583,2597,2604,
2611,2618,2632,2639,2646,2653,2667,2674,2681,2688,2702,2709,2716,2723,
2737,2744,2751,2758,2772,2779,2786,2793,2807,2814,2821,2828,2842,2849,
2856,2863,2877,2884,2891,2898,2912,2919,2926,2933,2947,2954,2961,2968,
2982,2989,2996,3003,3017,3024,3031,3038,3052,3059,3066,3073,3087,3094,
3101,3108,3122,3129,3136,3143,3157,3164,3171,3178,3192,3199
```

2.Write a program that accepts a sentence and calculate the number of letters and digits and count the occurrence of the characters and store them in dictionary.

```
sentence = input("Enter a sentence: ")

letters = 0
digits = 0
char_count = {}

for char in sentence:
    if char.isalpha():
        letters += 1
    elif char.isdigit():
        digits += 1
    if char in char_count:
        char_count[char] += 1
    else:
        char_count[char] = 1

print("Letters:", letters)
print("Digits:", digits)
print("Character Occurrences:", char_count)
```

```
Enter a sentence:  chethancheth63@gmail.com

Letters: 20
Digits: 2
Character Occurrences: {'c': 3, 'h': 4, 'e': 2, 't': 2, 'a': 2, 'n':
1, '6': 1, '3': 1, '@': 1, 'g': 1, 'm': 2, 'i': 1, 'l': 1, '.': 1,
'o': 1}
```

1. Use a list comprehension to square each odd number in a list. The list is input by a sequence of comma-separated numbers.

```
numbers = input("Enter comma-separated numbers: ")
nums = [int(x) for x in numbers.split(",")]
odds = [x for x in nums if x % 2 != 0]
squared_odds = [x**2 for x in odds]

print("Odd numbers:", odds)
print("Squares of odd numbers:", squared_odds)
```

```
Enter comma-separated numbers:  1,2,3,4,5,6,7,8,9

Odd numbers: [1, 3, 5, 7, 9]
Squares of odd numbers: [1, 9, 25, 49, 81]
```

1. Define a function which can generate a list where the values are squre of numbers between 1 and 20 (both included). Then the function needs to print the last 5 elements in the list

```
def print_last_five_squares():
    squares = [i**2 for i in range(1, 21)]
    print(squares[-5:])

print_last_five_squares()
```

```
[256, 289, 324, 361, 400]
```

1. Write a program which can map() and filter() to make a list whose elements are squre of even number in [6,9,10,12,15,30,25,81,100]

```
def square_even():
    numbers = [6,9,10,12,15,30,25,81,100]
    even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
    squared = list(map(lambda x: x ** 2, even_numbers))
    print(squared)

square_even()
```

```
[36, 100, 144, 900, 10000]
```