

Q2- Raman owns an old library in San Jose. He is not much inclined to technology and most the work at his library is completely manual. Tired of all the laborious manual work, he now wants to automate most of the work at library. You are a freelance software developer; Raman asks you to build a Library Management System for his library. As Raman is not much inclined towards technology, his expectations are very unclear, as a result you do not have the detailed set of requirements for the Library Management System. Now, answer the following question-

1. Which development strategy will you choose and why?

(Min 200 words, 2 Marks)

Answer:-

The hybrid model, which combines the use of Agile and Waterfall, is the one I would pick.

I am attempting to automate as much of Raman's manual work as possible. First, I'll separate different kinds of books, periodicals, and articles, then create a prototype to combine everything into one program, testing the viability of creating a library management system.

As soon as the prototype is finished, I will compile all the fundamental specifications (basic functionalities) for resolving the issue and make a realistic time estimate. By doing this, I can create a working base application with minimal time and capital.

I'll release it as a beta application for Raman to test and provide comments after I've finished developing the base product. This is primarily done to learn how the product works and to fix any bugs in the program.

I will release the actual application to Raman once the pilot program is finished, taking into account that the bugs have been fixed.

I would have wasted more time and money if I had used the Waterfall method in this case without first ensuring that the client could utilize the product and that it would function as intended. Therefore, I would advise using an agile development method

as it is the quickest and most effective way to finish the functionality and conduct the trial.

Now that the product has reached a stable state, I will switch to the waterfall model. I am now aware that the product is reliable and functional. I will compile Raman's requests for the next level of customization and incorporate the necessary features into the program. In order to accommodate the enormous amount of data, I will scale up the memory as the number of books rises.

Q3 - What differences has the web made to the field of software engineering?

Answer:-

Software engineering is a branch of engineering that deals with every aspect of creating software, from the early phases of system specification to maintaining the system after it has been put into use. Now the Web can be used as a platform for running applications, businesses are increasingly creating web-based platforms rather than local ones.

Web services make it possible to access program functionality from anywhere in the world. By default, we must see our developments as global, a strategy to provide computer services known as "cloud computing" involves having applications run remotely on the "cloud." Users pay according to use rather than purchasing software. The most common method for building web-based systems is software reuse. When creating these systems, you consider how to put them together using already-built software elements and systems.

Web-based systems should be created and provided incrementally and iteratively. It is now well acknowledged that it is impractical to establish all the criteria for such systems in advance. The abilities of web browsers place limitations on user interfaces. Rich user interfaces can be produced inside a web browser owing to technologies like AJAX, yet they are still challenging to use. The use of web forms with local scripting is more prevalent. Thus, the web has made software services available and made it possible to create widely dispersed service-based systems. Programming language and software reuse have significantly improved as a result of web-based systems development.

Q4. According to you,

1. What are two most important OOP design principles? Explain them?

2. Why is object-oriented programming better than procedural programming?

Clarification: Part 1: There may be many OOP design principles or concepts, you must explain any 2.

Answer:-

The two most important Object Oriented principles according to me are as follows:

Encapsulation: One of the pillars of OOP is encapsulation (object-oriented programming). It describes combining data with the processes that use that data. In order to prevent unauthorized parties from having direct access to the contents or state of a structured data object, encapsulation is utilized to conceal them inside of a class. The client classes call this method to obtain and modify the values inside the object. Publicly accessible methods, also known as getters and setters, are typically provided in the class to access the values. One of the most concrete examples of encapsulation is a school backpack. Our books, pencils, and other supplies can be kept in our school bags like that of "encapsulating" inside a class.

Inheritance: It is another crucial pillar of OOP. It is a feature that allows a class to take on the traits and qualities of another class. Because the derived class or the child class can reuse the base class's members by inheriting them, inheritance enables code reuse. To grasp the idea of inheritance, think of a practical illustration. The capacity to speak, move, eat and other traits are among those that children inherit from their parents. But not just his parents specifically inherited these qualities. His parents derive these characteristics from a different class known as mammals. These traits are again derived by this mammal class from the animal class. Similar principles apply to inheritance. According to the chosen visibility option, the data members of the base class are copied during inheritance and can be accessed in the derived class. Always in decreasing order, that is, from public to protected, is the order of accessibility. Inheritance comes in primarily five forms, they are as follows:

- ★ Single Inheritance
- ★ Multiple Inheritance
- ★ Multilevel Inheritance
- ★ Hierarchical Inheritance

★ Hybrid Inheritance

Even the most intelligent program built in the most unique paradigm eventually turns into a bundle of machine code shoving data into and out of memory and processor registers. Technically, they are all equal. What can be solved effectively and with ease of maintenance is always the question, not what can be solved using a particular approach. The main benefit of using an "Object-Oriented" approach to code structure is that you should have a variety of small pieces of knowledge, or objects, that are each clearly defined and well-encapsulated, and which are then combined into larger structures in specific ways to achieve the functionality you one requires, like that of abstraction, inheritance, polymorphism, etc.

However, "Procedural Programming" sees methods that work on the data of an object as the same, whereas in OOP, modules and procedures that operate on data are treated as independent entities. This is a key difference between procedural and object-oriented programming. Additionally, data in PP is susceptible to attack, which makes it less useful for resolving practical issues. The procedural programming approach also produces complex programs.

Q6. Implement the following problems in C++ and attach the program and the output

screen shot as a jpeg/png image.

Write a C++ program that asks the user for a source string, and a target string to find in

the source string. Write a function with two parameters - the source, and the target string

- that returns a boolean, if the target string is found in the source string. Use only std:string. Call this function from main() and print out all the strings for verification. Your

code will also be judged on algorithmic complexity.

Sample

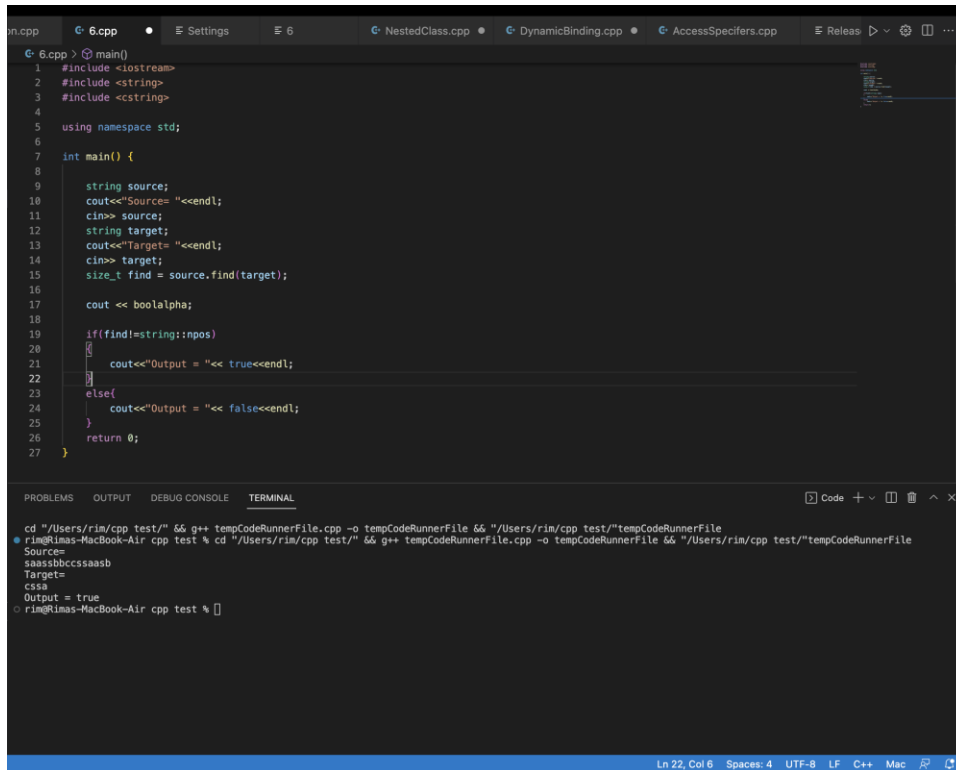
input: source = "saassbbccssaasb", target = "cssa"

Answer:-

```
#include <iostream>
```

```
#include <string>
```

```
#include <cstring>
using namespace std;
int main() {
    string source;
    cout<<"Source= "<<endl;
    cin>> source;
    string target;
    cout<<"Target= "<<endl;
    cin>> target;
    size_t find = source.find(target);
    cout << boolalpha;
    if(find!=string::npos)
    {
        cout<<"Output = "<< true<<endl;
    }
    else{
        cout<<"Output = "<< false<<endl;
    }
    return 0;
}
```



```
1 #include <iostream>
2 #include <string>
3 #include <string>
4
5 using namespace std;
6
7 int main() {
8     string source;
9     cout<<"Source= "<<endl;
10    cin>> source;
11    string target;
12    cout<<"Target= "<<endl;
13    cin>> target;
14    size_t find = source.find(target);
15
16    cout << boolalpha;
17
18    if(find!=string::npos)
19    {
20        cout<<"Output = "<< true<<endl;
21    }
22    else{
23        cout<<"Output = "<< false<<endl;
24    }
25    return 0;
26 }
```

```
cd "/Users/rin/cpp test/" && g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile && "/Users/rin/cpp test/"tempCodeRunnerFile
• rin@Rimas-MacBook-Air cpp test % cd "/Users/rin/cpp test/" && g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile && "/Users/rin/cpp test/"tempCodeRunnerFile
Source=
saashbccsaasb
Target=
cssa
Output = true
• rin@Rimas-MacBook-Air cpp test %
```

Q8. As a C++ programmer, answer the following questions - (2 Marks)

- 1. What is the difference between structure and class? Illustrate using an example. (100 words, 1 mark)**
- 2. Why is it recommended to define a member function outside the class? (100 words, 1 mark)**

Answer:-

Although structures and classes are nearly identical in terms of technicality, there are still significant variations. The primary distinction is that a class allows for the flexible combination of data and methods (functions), as well as the reusability known as inheritance. Data should normally be grouped using structures. The tiny differences in member default visibility are what account for the technical variance. Some of the differences between class and structure are shown here. Class is pass-by-reference and structure is pass-by-copy, which implies that although structure is a value type and its object is produced on the stack memory, Class is a reference type, and its object is formed on the heap memory. Structure does not support inheritance, whereas a class can build a subclass that would inherit the original class's properties and methods. By default, a class's members are all private whereas, structure is a

class whose members are by default made public. Classes enable garbage collector cleanup to be done before an item is deallocated because the garbage collector uses heap memory. When an instance is no longer used by another piece of code, objects are often deallocated. Garbage collection cannot be used on structures, resulting in ineffective memory management.

Class	Structure
<pre>class First { int a; //Private by default public: int f() { return a = 10; }; // Initializing to public };</pre>	<pre>struct Y { int f() { return a = 10; }; // Public by default private: int a; //Initializing to private };</pre>

The definition of a complex member function in a class makes it difficult to read and too complicated. Consider that a member function does not need to be defined within the class when it is declared using the function specifier inline. Note that even while a member function of a class declared within the class may get non-inline calls from the compiler, all other requirements for the function will remain the same. For instance, many compilation units can contain the function definition together with the class definition. Thus, one can distinguish between the interface and its realization by defining a member function outside of a class.