**Programing Language**

1. **Could you explain some concepts (Object, Instance, Class, Method, Attribute, Encapsulation, Inheritance, Polymorphism, Template)?**

* Object is an software entity which includes attributes and methods, a particular object is called an instance.
* Class which is a blueprint or prototype defines all common attributes and methods for objects which are have the same type. Class is essentially an Abstraction Data Type.
* Attribute is data used to describe properties (variables) of a object
* Method are behaviors which object can do. Each method have ability to effect attribute of object.
* Encapsulation is the mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse.
* Inheritance is the process by which one object can acquire the properties of another object. This is important because it supports the concept of classification.
* Polymorphism is the ability of objects belonging to different data types to respond to method calls of methods of the same name, each one according to an appropriate type­specific behavior.

1. **Could you give an C++ example and show concepts in that example?**

|  |
| --- |
| 1. class *person* 2. { 3. *string* first\_name; 4. *string* last\_name; 5. *string* gender; 6. int age; 7. public: 8. person(); 9. person(*string*); 10. person(*string*, *string*, int); 11. *string* get\_fistname(); 12. *string* get\_fullname(); 13. int get\_age(); 14. void set\_firstname(*string*); 15. void set\_lastname(*string*); 16. void set\_gender(*string*); 17. void set\_age(int); 18. }; |

* Declare a class: right here class named person and person is a object.

|  |
| --- |
| * class *student*: public *person* * { * *person* X; * *string* id\_student; * *string* major; * float gpa; * public: * student(); * student(*string*, int); * student(*person*, *string* , *string* , float); * *string* show\_info(); * }; |

* Class name is student, class person inherited by class student

1. **Could you explain some C++ features (Template, Friend function, Friend class, Function overloading, Operator overloading, Virtual function)?**

* Templates are a feature of the C++ programming language that allows functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one.
* A friend function is a function that is not a member of a class but has access to the class's private and protected members. Friend functions are not considered class members.
* A friend class is a class all of whose member functions are friend functions of a class.
* C++ allows specification of more than one function of the same name in the same scope. This is called Function Overloading. Overloaded functions must be different from input argument list.
* A virtual function is a member function that you expect to be redefined in derived classes.

1. **When do you use Vector, List, Set, Map, Bitset?**

* C++ vectors are sequence containers that store elements. Specifically used to work with dynamic data, C++ vectors may expand depending on the elements they contain.
* Lists are sequence containers that allow non-contiguous memory allocation. As compared to vector, list has slow traversal, but once a position has been found, insertion and deletion are quick. For implementing a singly linked list, we use forward list.
* Sets are frequently used containers in competitive programming; whenever there is a need of storing the elements in a sorted manner
* Map is dictionary like data structure. It is a sequence of (key, value) pair, where only single value is associated with each unique key. It is often referred as associative array. In map key values generally used to sort the elements.
* The BitSet class creates a special type of array that holds bit values. The BitSet array can increase in size as needed. This makes it similar to a vector of bits.

1. **Could you give an C++ example by using all basic features?**

|  |
| --- |
| 1. class *A* 2. { 3. private: 4. int numA; 5. friend int add(*A*, *B*); // friend function declaration 6. public: 7. A() : numA(12) {}    //initialize numA to 12 8. void operator+() 9. { 10. numA++; 11. cout << "increment" << numA << endl; 12. } 13. }; |

* Friend function and operator overloading in class A

|  |
| --- |
| * class *B* * { * private: * int numB; * friend int add(*A*, *B*);  // friend function declaration * public: * B() : numB(1) {}   //initialize numB to 1 * void operator-() * { * numB--; * cout << "Decrement: " << numB <<endl; * } * }; |

* Friend function and operator overloading in class A

|  |
| --- |
| * int add(*A* *objectA*, *B* *objectB*) * { * return (*objectA*.numA + *objectB*.numB); * } |

* Access members of both classes by Friend function.

|  |
| --- |
| * int add(int *a*, int *b*, int *c*)   //Function overloading * { * return *a*+*b*+*c*; * } * float add(float *a*, int *b*, int *c*, float *d*) //Function overloading * { * return *a*+*b*+*c*+*d*; * } |

* Function overloading.