1. What is function overloading and operator overloading?

   Function overloading: C++ enables several functions of the same name to be defined, as long as these functions have different sets of parameters (at least as far as their types are concerned). This capability is called function overloading. When an overloaded function is called, the C++ compiler selects the proper function by examining the number, types and order of the arguments in the call. Function overloading is commonly used to create several functions of the same name that perform similar tasks but on different data types.

   Operator overloading allows existing C++ operators to be redefined so that they work on objects of user-defined classes. Overloaded operators are syntactic sugar for equivalent function calls. They form a pleasant facade that doesn't add anything fundamental to the language (but they can improve understandability and reduce maintenance costs).

2. Explain what is a reference variable in C++?

   A reference variable is just like a pointer with few differences. It is declared using & Operator. In other words, reference is another name for an already existing variable.

3. What is the need /use of function overloading?

   In function overloading, the function has single name but when it called from main function by passing different types of arguments, then it behaves according to the passed from the function.

   The function different actions based on arguments, hence it's called as function overloading.

   The need is that

   1. Rewriting of the code is not required
   2. Memory is saved.

4. What are the c++ tokens?

   c++ has the following tokens

   I. keywords

   II. Identifiers

   III. Constants

   IV. Strings

   V. operators

5. What do you mean by implicit conversion?

   Whenever data types are mixed in an expression then c++ performs the conversion automatically.

   Here smaller type is converted to wider type.

   Example- in case of integer and float integer is converted into float type.

6. What are the different ways of passing parameters to the functions? Which to use when?

Call by value: In this method, only the values are passed to the function as arguments. In this case, the values passed will be manipulated within the program, but its actual value in the variable will not be modified.

Call by address: Here address of the actual parameters is passed instead of values. One can opt for this method if one needs to modify both actual parameters and formal parameters.

Call by reference: The actual parameters are receives new reference variables as formal parameters. One can opt this method if they want both actual parameters and formal parameters to be modified.

7. What does extern mean in a function declaration?

If we need to use a function outside the file in which it is defined, we can declare it using the keyword extern. This makes a variable, function definition, makes it to be used within the source file as well as in other files too. It does not alter the definition of the variable or function. If there is any variable or function declared as extern already exists in the file scope without extern, and its reference is for such variable or function within the file itself. If it does not exists, then it searches for any extern variable or function in other files.

8. How do we initialize a pointer to a function?

Pointer to a function can be initialized in following way:

```
#include
usingnamespacestd;
voidfunc(intx)
{
}
voidmain()
{
 void(*fp)(int);
 fp=func;
 fp(1);
}
```

9. What is a friend function?

Private data members of the class are not accessible by other class or outside the class. Similarly, protected members of the class are not accessible outside the class. Sometimes these data members need to be accessed by other classes or in the derived classes. In such cases, if we make the function as friend of the class, its private and protected data members can be accessed.

10. Do inline functions improve performance? Explain.

    Whenever an inline function is called, the function call is directly replaced by the function definition in the program. Hence the overhead of stopping the execution at function call and jumping to execute the function and then coming back to the calling function is not there here. It also reduces the space as there is no separate set of instruction to handle the function is written in the memory.