OBJECTIVE
Implement class and object concepts and function overloading.

PROGRAM

C++ allows you to specify more than one definition for a **function** name or an **operator** in the same scope, which is called **function overloading** and **operator overloading** respectively.
An overloaded declaration is a declaration that is declared with the same name as a previously declared declaration in the same scope, except that both declarations have different arguments and obviously different definition (implementation).
When you call an overloaded **function** or **operator**, the compiler determines the most appropriate definition to use, by comparing the argument types you have used to call the function or operator with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called **overload resolution**.
Function Overloading in C++:
You can have multiple definitions for the same function name in the same scope. The definition of the function must differ from each other by the types and/or the number of arguments in the argument list. You cannot overload function declarations that differ only by return type.
Following is the example where same function **print()** is being used to print different data types:

```cpp
#include <iostream>
using namespace std;
class printData {
   public:
      void print(int i) {
       cout << "Printing int: " << i << endl;
      }
      void print(double  f) {
       cout << "Printing float: " << f << endl;
      }
      void print(char* c) {
       cout << "Printing character: " << c << endl;
      }
};

int main(void) {
   printData pd;

   // Call print to print integer
   pd.print(5);
```

```cpp
    // Call print to print float
    pd.print(500.263);
       // Call print to print character
    pd.print("Hello C++");

    return 0;
}
```

Output:
Printing int: 5
Printing float: 500.263
Printing character: Hello C++