FUNCTION CALL OPERATOR OVERLOADING IN C++

The function call operator can be overloaded for objects of class type. When you overload , you are not creating a new way to call a function. Rather, you are creating an operator function that can be passed an arbitrary number of parameters.

Following example explains how a function call operator can be overloaded.

```
#include <iostream>
using namespace std;

class Distance
{
   private:
      int feet;             // 0 to infinite
      int inches;           // 0 to 12
   public:
      // required constructors
      Distance(){
         feet = 0;
         inches = 0;
      }
      Distance(int f, int i){
         feet = f;
         inches = i;
      }
      // overload function call
      Distance operator()(int a, int b, int c)
      {
         Distance D;
         // just put random calculation
         D.feet = a + c + 10;
         D.inches = b + c + 100 ;
         return D;
      }
      // method to display distance
      void displayDistance()
      {
         cout << "F: " << feet <<  " I:" <<  inches << endl;
      }

};
int main()
{
   Distance D1(11, 10), D2;

   cout << "First Distance : ";
   D1.displayDistance();

   D2 = D1(10, 10, 10); // invoke operator()
   cout << "Second Distance :";
   D2.displayDistance();

   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
First Distance : F: 11 I:10
Second Distance :F: 30 I:120
```