Experiment 7

AIM:

Write a program to perform different arithmetic operations like +, -, *, / using inline functions.

Theory:

If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time. Unlike macro substitution there is error checking and hence it more safe.

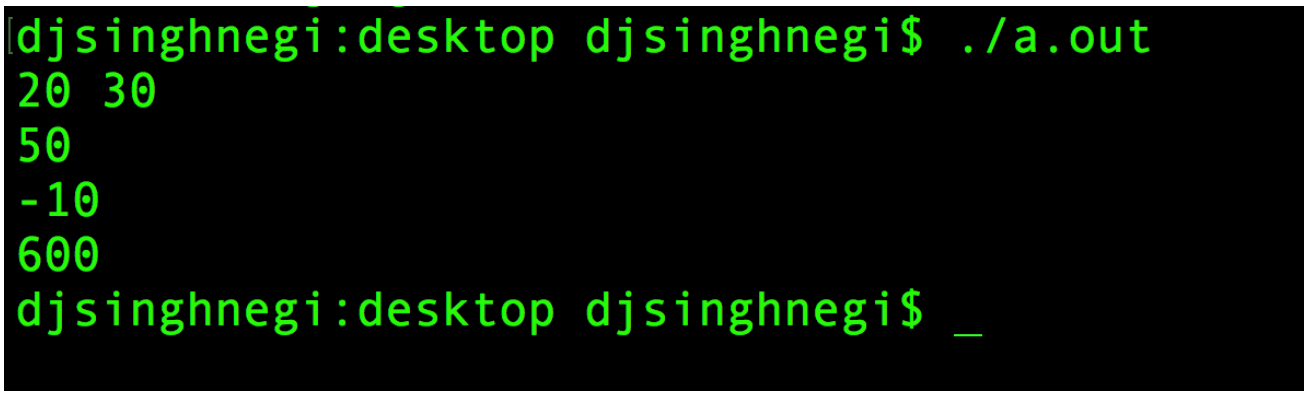
Compiler can ignore the request for inlining. Compiler may not perform inlining in such circumstances like:

1. If a function contains a loop. (for, while, do-while).
2. If a function contains static variables.
3. If a function is recursive.
4. If a function return type is other than void, and the return statement doesn't exist in function body.
5. If a function contains switch or goto statement.

Code:

```
#include <iostream>
using namespace std;
class Operation {
public:
    inline int add(int x, int y) {
        return x + y;
    }
    inline int subtract(int x, int y) {
        return x - y;
    }
    inline int multiply(int x, int y) {
        return x * y;
    }
    inline int divide(int x, int y) {
        return x / y;
    }
};
int main() {
    Operation op;
    int x, y;
    cin >> x >> y;
    cout << op.add(x, y) << endl;
    cout << op.subtract(x, y) << endl;
    cout << op.multiply(x, y) << endl;
}
```

Output:



Discussion:

The inline functions add, subtract, multiply, divide all work like normal functions.