Experiment 9

AIM:

Write a program to string operations using operator overloading:

1. =, string copy
2. ==, >, <, comparision
3. +, concatenation

Theory:

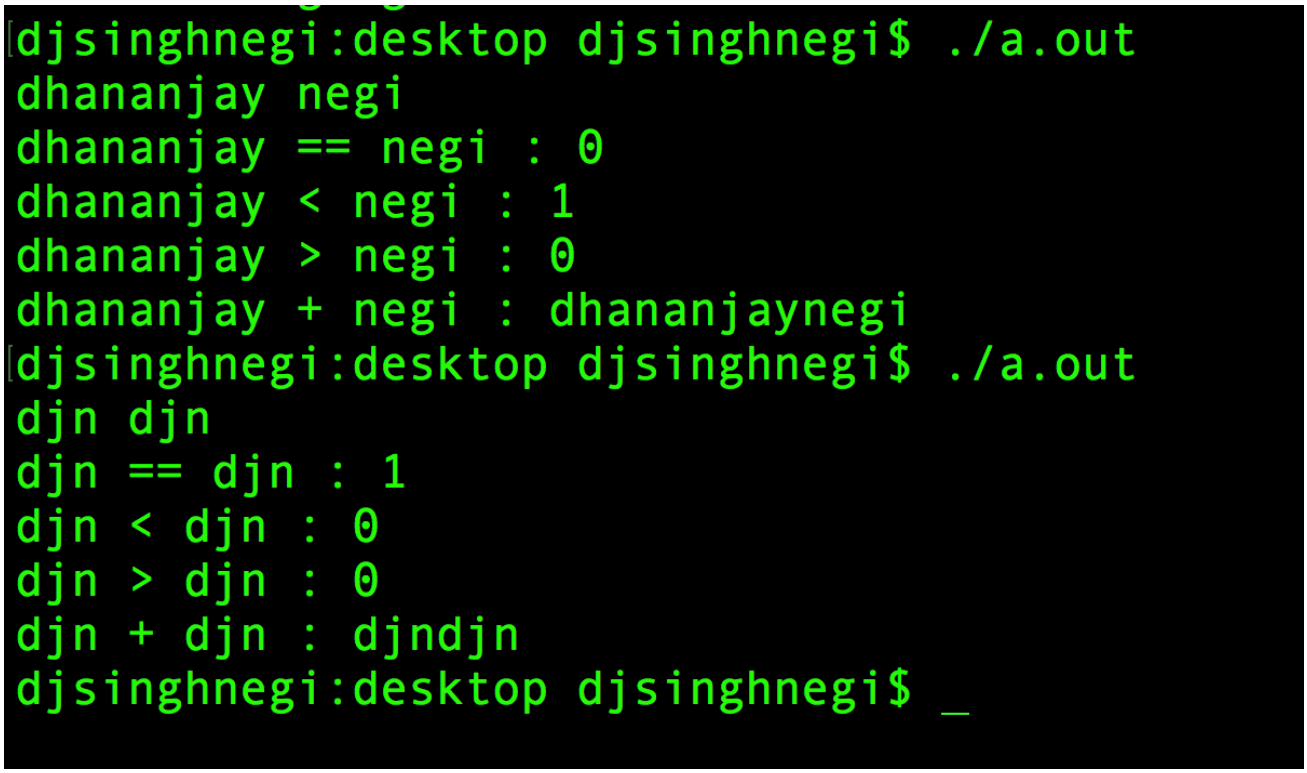
C++ allows you to specify more than one definition for an operator in the same scope, which is called operator overloading. An overloaded declaration is a declaration that is declared with the same name as a previously declared declaration in the same scope, except that both declarations have different arguments and obviously different definition (implementation). You can redefine or overload most of the built-in operators available in C++. Thus, a programmer can use operators with user-defined types as well. Overloaded operators are functions with special names: the keyword "operator" followed by the symbol for the operator being defined. Like any other function, an overloaded operator has a return type and a parameter list.

Code:

```
#include <iostream>
#include < cstring >
using namespace std;
class String {
    char * s;
    int len;
    int cmp(const String & a,
        const String & b) {
        int i = 0, j = 0;
        while (i < a.len && j < b.len) {
            if (a.s[i] < b.s[j])
              return -1;
            else if (a.s[i] > b.s[j])
        return 1;
            i++;
            j++;
        }
        if (a.len < b.len)
        return -1;
        else if (a.len > b.len)
        return 1;
        else
        return 0;
    }
    public: String() {
        s = '\0';
        len = 0;
    }
    String(char * p) {
        len = strlen(p);
        s = new char[len + 1];
        strcpy(s, p);
    }
```

```
    void operator = (const String & a) {
        delete s;
        len = a.len;
        s = new char[len + 1];
        s = strcpy(s, a.s);
    }
    bool operator < (const String & a) {
        return cmp( * this, a) == -1;
    }
    bool operator > (const String & a) {
        return cmp( * this, a) == 1;
    }
    bool operator == (const String & a) {
        return cmp( * this, a) == 0;
    }
    String operator + (const String & b) {
        char * n = new char[len + b.len + 1];
        strcpy(n, s);
        strcat(n, b.s);
        String ans = String(n);
        delete[] n;
        return ans;
    }
    friend ostream & operator << (ostream & , String & );
    friend istream & operator >> (istream & , String & );
};
ostream & operator << (ostream & op, String & a) {
    op << a.s;
    return op;
}
istream & operator >> (istream & op, String & a) {
    char temp[100];
    cin >> temp;
    a = String(temp);
    return op;
}
int main() {
    String a, b, c;
    cin >> a >> b;
    cout << a << " == " << b << " : " << (a == b) << endl;
    cout << a << " < " << b << " : " << (a < b) << endl;
    cout << a << " > " << b << " : " << (a > b) << endl;
    c = a + b;
    cout << a << " + " << b << " : " << c << endl;
    return 0;
}
```

Output:



Discussion:

The operators +,=,==,< and > have been overloaded for string operations.