**Experiment 8**

**AIM:**

Write a program to string operations using operator overloading:

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

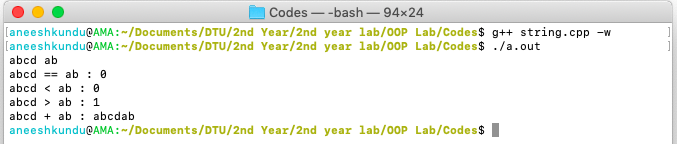
**Theory:**

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:**

1. #include < iostream >
2. #include < cstring >
3. using namespace std;
4. class String {
5. char \* s;
6. int len;
7. int cmp(const String & a,
8. const String & b) {
9. int i = 0, j = 0;
10. while (i < a.len && j < b.len) {
11. if (a.s[i] < b.s[j])
12. return -1;
13. else if (a.s[i] > b.s[j])
14. return 1;
15. i++;
16. j++;
17. }
18. if (a.len < b.len)
19. return -1;
20. else if (a.len > b.len)
21. return 1;
22. else
23. return 0;
24. }
25. public: String() {
26. s = '\0';
27. len = 0;
28. }
29. String(char \* p) {
30. len = strlen(p);
31. s = new char[len + 1];
32. strcpy(s, p);
33. }
34. void operator = (const String & a) {
35. delete s;
36. len = a.len;
37. s = new char[len + 1];
38. s = strcpy(s, a.s);
39. }
40. bool operator < (const String & a) {
41. return cmp( \* this, a) == -1;
42. }
43. bool operator > (const String & a) {
44. return cmp( \* this, a) == 1;
45. }
46. bool operator == (const String & a) {
47. return cmp( \* this, a) == 0;
48. }
49. String operator + (const String & b) {
50. char \* n = new char[len + b.len + 1];
51. strcpy(n, s);
52. strcat(n, b.s);
53. String ans = String(n);
54. delete[] n;
55. return ans;
56. }
57. friend ostream & operator << (ostream & , String & );
58. friend istream & operator >> (istream & , String & );
59. };
60. ostream & operator << (ostream & op, String & a) {
61. op << a.s;
62. return op;
63. }
64. istream & operator >> (istream & op, String & a) {
65. char temp[100];
66. cin >> temp;
67. a = String(temp);
68. return op;
69. }
70. int main() {
71. String a, b, c;
72. cin >> a >> b;
73. cout << a << " == " << b << " : " << (a == b) << endl;
74. cout << a << " < " << b << " : " << (a < b) << endl;
75. cout << a << " > " << b << " : " << (a > b) << endl;
76. c = a + b;
77. cout << a << " + " << b << " : " << c << endl;
78. return 0;
79. }

**Output:**

****

**Discussion:**

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