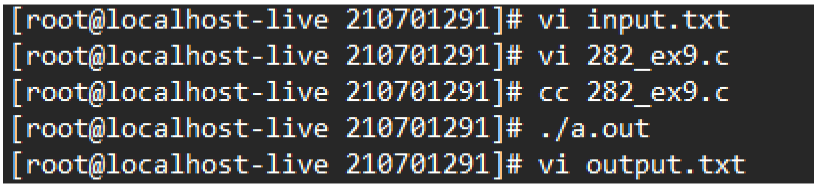
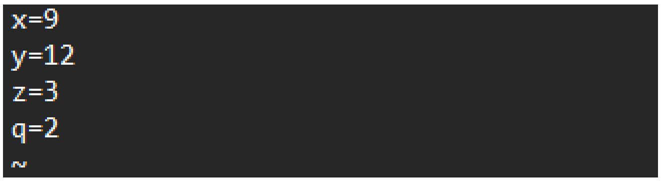
```
                        case '+':
                            interrslt=(s[2]-48)+(s[4]-48);
                            break;
                        case '-':
                            interrslt=(s[2]-48)-(s[4]-48);
                            break;
                        case '*':
                            interrslt=(s[2]-48)*(s[4]-48);
                            break;
                        case '/':
                            interrslt=(s[2]-48)/(s[4]-48);
                            break;
                        default:
                            interrslt = 0;
                            break; }
                    fprintf(fp2,"/*Constant Folding*/\n");
                    fprintf(fp2,"%c = %lf\n",result,interrslt);
                    flag2 = 0;
                }
            } else {
                fprintf(fp2,"Not Optimized\n");
                fprintf(fp2,"%s\n",s);
            }
        } else {
            fprintf(fp2,"%s\n",s);
        }
        fscanf(fp1,"%s",s);
    }
    fclose(fp1);
    fclose(fp2);
}
```

OUTPUT:



```
[root@localhost-live 210701291]# vi input.txt
[root@localhost-live 210701291]# vi 282_ex9.c
[root@localhost-live 210701291]# cc 282_ex9.c
[root@localhost-live 210701291]# ./a.out
[root@localhost-live 210701291]# vi output.txt
```

//output.txt



```
x=9
y=12
z=3
q=2
~
```

RESULT:

Ex No: 10

Date:

IMPLEMENT CODE OPTIMIZATION TECHNIQUES
DEAD CODE AND COMMON SUB EXPRESSION ELIMINATION

AIM:

To write a C program to implement the dead code elimination and common subexpression elimination (code optimization) techniques.

ALGORITHM:

- Start
- Create the input file which contains three address code.
- Open the file in read mode.
- If the file pointer returns NULL, exit the program else go to 5.
- Scan the input symbol from left to right.
- Store the first expression in a string.
- Compare the string with the other expressions in the file.
- If there is a match, remove the expression from the input file.
- Perform these steps 5-8 for all the input symbols in the file.
- Scan the input symbol from the file from left to right.
- Get the operand before the operator from the three address code.
- Check whether the operand is used in any other expression in the three address code.
- If the operand is not used, then eliminate the complete expression from the three-address code else go to 14.
- Perform steps 11 to 13 for all the operands in the three address code till end of the file is reached.
- Stop.

PROGRAM:

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

 struct op
{
   char l;
   char r[20];
  }
```