ASSIGNMENT: 06

Name: Abhijeet Biswas
SRN: 201900400
Roll No: 05
Div: B

Question:

Implement the following code optimizations on the input 3-address code in the form of quadruples:

a) Common subexpression elimination

b) Constant folding

Output : Optimized 3 - address code

Input:

1. Code Optimization

```
1: + y z t1
2: = t1 x
3: + b c t2
4: = t2 a
5: + y z t3
6: = x b
7: = t3 j
8: + b c t4
9: = t4 d
10: * b c t5
11: = t5 f
12: + y z t6
13: = t6 g
```

2. Code Folding and Propogation

```
1: = 30 c
2: + y z t1
3: = t1 x
4: + b c t2
5: = t2 a
6: + y z t3
7: = t3 j
8: = x b
9: = 20 b
10: + b c t5
11: = t4 d
12: * b c t5
13: = t5 f
14: + y z t6
15: = t6 g
```

Code:

Ass6.py –

```
class CodeOptimization():
    quadrapleTable = []
    noOfStatements = 0
    result = []
    constants = {}
    def takeInput(self) -> None:
        state = []
        self.noOfStatements = int(input('\nNo. of Statements: '))
        print('\n+-----+----------+------+------+--------+')
        for i in range(self.noOfStatements):
            a = input(f"{i+1}: ")
            state.append(a)
        self.makeQudraple(state)

    def makeQudraple(self,inputStates:list) -> None:
        self.noOfStatements = len(inputStates)
        for n in range(self.noOfStatements):
            var = inputStates[n].split(' ')
            if len(var) == 4:
                self.quadrapleTable.append([n+1, var[0], var[1],
var[2], var[3]])
            elif len(var)== 3:
                self.quadrapleTable.append([n+1, var[0], var[1], "  ",
var[2]])
                if var[1].isnumeric():
```

```
                                self.constants.update({var[2]:[var[1],n]})


        def optimize(self) -> None:
                self.result.append(self.quadrapleTable[0][4])
                n=1
                self.result.clear()
                self.result.append(self.quadrapleTable[0][4])
                while(1):
                        if n >= self.noOfStatements:
                                return
                        for i in range(n-1,-1,-1):
                                if (self.quadrapleTable[n][1] ==
self.quadrapleTable[i][1] and
                                        (self.quadrapleTable[n][2] ==
self.quadrapleTable[i][2] or self.quadrapleTable[n][2] ==
self.quadrapleTable[i][3]) and
                                        (self.quadrapleTable[n][3] ==
self.quadrapleTable[i][3] or self.quadrapleTable[n][2] ==
self.quadrapleTable[i][2])):


                                        if self.quadrapleTable[n][2] not in self.result
and self.quadrapleTable[n][3] != self.quadrapleTable[i][4]:
                                                print(f'\n\nOptimizing at State: {n+1}')
                                                print('\n+-----+----------+------+------+-----
---+')
                                                self.quadrapleTable.pop(n)
                                                self.noOfStatements-=1
                                                self.quadrapleTable[n][2] =
self.quadrapleTable[i][4]
```

```
                                        s.printQuadTable()
                    self.result.append(self.quadrapleTable[n][4])
                    n+=1


    def constantFoldingAndPropogation(self):
            self.result.clear()
            self.result.append(self.quadrapleTable[0][4])
            n=1
            while(1):
                if n >= self.noOfStatements:
                    return
                for i in range(n-1,-1,-1):
                    if self.quadrapleTable[i][2] in self.constants.keys()
and i >= self.constants.get(self.quadrapleTable[i][2])[1]:
                                    self.quadrapleTable[i][2] =
str(self.constants.get(self.quadrapleTable[i][2])[0])
                    if self.quadrapleTable[i][3] in self.constants.keys()
and i >= self.constants.get(self.quadrapleTable[i][3])[1]:
                                    self.quadrapleTable[i][3] =
str(self.constants.get(self.quadrapleTable[i][3])[0])
                    if self.quadrapleTable[i][1] in ['+','-','/','*','%'] and
self.quadrapleTable[i][2].isnumeric() and self.quadrapleTable[i][3].isnumeric():
                                    if self.quadrapleTable[i][1] == '+':
                                        sum = int(self.quadrapleTable[i][2]) +
int(self.quadrapleTable[i][3])

    self.quadrapleTable.remove(self.quadrapleTable[i])
                                        self.quadrapleTable[i][2] = str(sum)
```

```
        self.constants.update({self.quadrapleTable[i][4]:[(self.quadrapleTable[i][
2]),i]})
                                        self.noOfStatements-=1
                                    if self.quadrapleTable[i][1] == '*':
                                        sum = int(self.quadrapleTable[i][2]) *
int(self.quadrapleTable[i][3])

        self.quadrapleTable.remove(self.quadrapleTable[i])
                                        self.quadrapleTable[i][2] = str(sum)

        self.constants.update({self.quadrapleTable[i][4]:[(self.quadrapleTable[i][
2]),i]})
                                        self.noOfStatements-=1
                                    if self.quadrapleTable[i][1] == '/':
                                        sum = int(self.quadrapleTable[i][2]) /
int(self.quadrapleTable[i][3])

        self.quadrapleTable.remove(self.quadrapleTable[i])
                                        self.quadrapleTable[i][2] = str(sum)

        self.constants.update({self.quadrapleTable[i][4]:[(self.quadrapleTable[i][
2]),i]})
                                        self.noOfStatements-=1
                                    if self.quadrapleTable[i][1] == '%':
                                        sum = int(self.quadrapleTable[i][2]) %
int(self.quadrapleTable[i][3])

        self.quadrapleTable.remove(self.quadrapleTable[i])
                                        self.quadrapleTable[i][2] = str(sum)
```

```
        self.constants.update({self.quadrapleTable[i][4]:[(self.quadrapleTable[i][
2]),i]})
                                        self.noOfStatements-=1
                        if self.quadrapleTable[i][2].isnumeric() and
self.quadrapleTable[i][1] == '=':

        self.constants.update({self.quadrapleTable[i][4]:[int(self.quadrapleTable[
i][2]),i]})
                                    break
                n+=1


        def printQuadTable(self) -> None:
            print('+-----+----------+------+------+--------+')
            print('| No. | Operator | Arg1 | Arg2 | Result |')
            print('+-----+----------+------+------+--------+')
            for n in range(self.noOfStatements):
                print("| {:<4}| {:<9}| {:<5}| {:<5}| {:<7}|".format(
                    n+1,
                    self.quadrapleTable[n][1],
                    self.quadrapleTable[n][2],
                    self.quadrapleTable[n][3],
                    self.quadrapleTable[n][4],
                ))
            print('+-----+----------+------+------+--------+')
            print("\n\n Constants:")
            for i in self.constants.keys():
                print(i,": ",self.constants.get(i)[0])
```

```
#__main__()
s = CodeOptimization()
print('\n+-----+----------+------+------+--------+')
print("\t1. Code Optimzation")
print("\n\t2. Code Folding and Propogation")
print('\n+-----+----------+------+------+--------+')
ch = int(input("\nEnter your choice (1-2): "))
print('\n+-----+----------+------+------+--------+')
if ch==1:
        s.takeInput()
        print("\nInput Table: ")
        print('\n+-----+----------+------+------+--------+')
        s.printQuadTable()
        s.optimize()
elif ch == 2:
        s.takeInput()
        print("\nInput Table: ")
        print('\n+-----+----------+------+------+--------+')
        s.printQuadTable()
        s.optimize()
        s.constantFoldingAndPropogation()
        s.printQuadTable()
else:
        print("\nEnter Valid Choice")
        print('\n+-----+----------+------+------+--------+')
        exit(1)
```

Output:

```
PS C:\Users\hp\Documents\VS Code's\TY\Sem VI\CD\Ass6> python -u "c:\Users\hp\Documents\V

+-----+----------+------+------+--------+
        1. Code Optimzation

        2. Code Folding and Propogation

+-----+----------+------+------+--------+

Enter your choice (1-2): 1

+-----+----------+------+------+--------+

No. of Statements: 13

+-----+----------+------+------+--------+
1: + y z t1
2: = t1 x
3: + b c t2
4: = t2 a
5: + y z t3
6: = x b
7: = t3 j
8: + b c t4
9: = t4 d
10: * b c t5
11: = t5 f
12: + y z t6
13: = t6 g

Input Table:

+-----+----------+------+------+--------+
+-----+----------+------+------+--------+
| No. | Operator | Arg1 | Arg2 | Result |
+-----+----------+------+------+--------+
| 1   | +        | y    | z    | t1     |
| 2   | =        | t1   |      | x      |
| 3   | +        | b    | c    | t2     |
| 4   | =        | t2   |      | a      |
| 5   | +        | y    | z    | t3     |
| 6   | =        | x    |      | b      |
| 7   | =        | t3   |      | j      |
| 8   | +        | b    | c    | t4     |
| 9   | =        | t4   |      | d      |
| 10  | *        | b    | c    | t5     |
| 11  | =        | t5   |      | f      |
| 12  | +        | y    | z    | t6     |
| 13  | =        | t6   |      | g      |
+-----+----------+------+------+--------+
```

```
 Constants:


Optimizing at State: 5

+-----+----------+------+------+--------+
+-----+----------+------+------+--------+
| No. | Operator | Arg1 | Arg2 | Result |
+-----+----------+------+------+--------+
| 1   | +        | y    | z    | t1     |
| 2   | =        | t1   |      | x      |
| 3   | +        | b    | c    | t2     |
| 4   | =        | t2   |      | a      |
| 5   | =        | t1   |      | b      |
| 6   | =        | t3   |      | j      |
| 7   | +        | b    | c    | t4     |
| 8   | =        | t4   |      | d      |
| 9   | *        | b    | c    | t5     |
| 10  | =        | t5   |      | f      |
| 11  | +        | y    | z    | t6     |
| 12  | =        | t6   |      | g      |
+-----+----------+------+------+--------+


 Constants:


Optimizing at State: 11

+-----+----------+------+------+--------+
+-----+----------+------+------+--------+
| No. | Operator | Arg1 | Arg2 | Result |
+-----+----------+------+------+--------+
| 1   | +        | y    | z    | t1     |
| 2   | =        | t1   |      | x      |
| 3   | +        | b    | c    | t2     |
| 4   | =        | t2   |      | a      |
| 5   | =        | t1   |      | b      |
| 6   | =        | t3   |      | j      |
| 7   | +        | b    | c    | t4     |
| 8   | =        | t4   |      | d      |
| 9   | *        | b    | c    | t5     |
| 10  | =        | t5   |      | f      |
| 11  | =        | t1   |      | g      |
+-----+----------+------+------+--------+
```

```
PS C:\Users\hp\Documents\VS Code's\TY\Sem VI\CD\Ass6> python -u "c:\Users\h

+-----+----------+------+------+--------+
        1. Code Optimzation

        2. Code Folding and Propogation

+-----+----------+------+------+--------+

Enter your choice (1-2): 2

+-----+----------+------+------+--------+

No. of Statements: 15

+-----+----------+------+------+--------+
1: = 30 c
2: + y z t1
3: = t1 x
4: + b c t2
5: = t2 a
6: + y z t3
7: = t3 j
8: = x b
9: = 20 b
10: + b c t5
11: = t4 d
12: * b c t5
13: = t5 f
14: + y z t6
15: = t6 g

Input Table:
```

Input Table:

No.	Operator	Arg1	Arg2	Result
1	=	30		c
2	+	y	z	t1
3	=	t1		x
4	+	b	c	t2
5	=	t2		a
6	+	y	z	t3
7	=	t3		j
8	=	x		b
9	=	20		b
10	+	b	c	t5
11	=	t4		d
12	*	b	c	t5
13	=	t5		f
14	+	y	z	t6
15	=	t6		g

Constants:
c : 30
b : 20

Optimizing at State: 6

No.	Operator	Arg1	Arg2	Result
1	=	30		c
2	+	y	z	t1
3	=	t1		x
4	+	b	c	t2
5	=	t2		a
6	=	t1		j
7	=	x		b
8	=	20		b
9	+	b	c	t5
10	=	t4		d
11	*	b	c	t5
12	=	t5		f
13	+	y	z	t6
14	=	t6		g

```
 Constants:
c :  30
b :  20


Optimizing at State: 13

+-----+----------+------+------+--------+
+-----+----------+------+------+--------+
| No. | Operator | Arg1 | Arg2 | Result |
+-----+----------+------+------+--------+
| 1   | =        | 30   |      | c      |
| 2   | +        | y    | z    | t1     |
| 3   | =        | t1   |      | x      |
| 4   | +        | b    | c    | t2     |
| 5   | =        | t2   |      | a      |
| 6   | =        | t1   |      | j      |
| 7   | =        | x    |      | b      |
| 8   | =        | 20   |      | b      |
| 9   | +        | b    | c    | t5     |
| 10  | =        | t4   |      | d      |
| 11  | *        | b    | c    | t5     |
| 12  | =        | t5   |      | f      |
| 13  | =        | t1   |      | g      |
+-----+----------+------+------+--------+


 Constants:
c :  30
b :  20
+-----+----------+------+------+--------+
| No. | Operator | Arg1 | Arg2 | Result |
+-----+----------+------+------+--------+
| 1   | =        | 30   |      | c      |
| 2   | +        | y    | z    | t1     |
| 3   | =        | t1   |      | x      |
| 4   | +        | b    | 30   | t2     |
| 5   | =        | t2   |      | a      |
| 6   | =        | t1   |      | j      |
| 7   | =        | x    |      | b      |
| 8   | =        | 20   |      | b      |
| 9   | =        | 50   |      | d      |
| 10  | =        | 600  |      | f      |
| 11  | =        | t1   |      | g      |
+-----+----------+------+------+--------+
```

```
 Constants:
c :  30
b :  20
d :  50
f :  600
```