**Practical No. 08**

**Name: Amarsingh Kashyap**

**Roll No: 101**

**Branch: CSE (3rd year)**

**Aim :** Write a code to implement Local optimization techniques until no further optimization is possible for the given three address code.

**CODE :**

**import** re  
  
lhs = []  
rhs = []  
  
**def** deadCode(lhs, rhs):  
 print()  
 LHS = []  
 RHS = []  
 sz = len(lhs)  
 **for** i **in** range(sz):  
 *# print(lhs[i])* l = lhs[i]  
 r = rhs[i]  
 **for** j **in** range(i + 1, sz):  
 *# print(rhs[j])* **if** rhs[j].find(l) >= 0:  
 *# print(rhs[j])* LHS.append(l)  
 RHS.append(r)  
 **break** LHS.append(lhs[sz - 2])  
 RHS.append(rhs[sz - 2])  
 LHS.append(lhs[sz - 1])  
 RHS.append(rhs[sz - 1])  
 print(**"======================================================================================"**)  
 print(**"After dead code elimination: "**)  
 **for** i **in** range(len(LHS)):  
 print(LHS[i] + **' : '** + RHS[i])  
  
 **return** LHS, RHS  
  
  
**def** constantProp(lhs, rhs):  
 print()  
 sz = len(lhs)  
 **for** i **in** range(sz):  
 **if** rhs[i].isdigit():  
 l = lhs[i]  
 r = rhs[i]  
 **for** j **in** range(i + 1, sz):  
 **if** lhs[j] == l:  
 **break  
 if** l **in** rhs[j]:  
 index = rhs[j].find(l)  
 k = rhs[j].split(l)  
 s = k[0] + r + k[1]  
 rhs[j] = s  
 print(**"======================================================================================"**)  
 print(**"After constant propogation: "**)  
 **for** i **in** range(len(lhs)):  
 print(lhs[i] + **' : '** + rhs[i])  
  
  
**def** copyProp(lhs, rhs):  
 print()  
 sz = len(lhs)  
 **for** i **in** range(sz):  
 **if** (**'+' not in** rhs[i]) **and** (**'-' not in** rhs[i]) **and** (**'\*' not in** rhs[i]) **and** (**'/' not in** rhs[i]) **and not** rhs[  
 i].isdigit():  
 l = lhs[i]  
 r = rhs[i]  
 **for** j **in** range(i + 1, sz):  
 **if** lhs[j] == l:  
 **break  
 if** l **in** rhs[j]:  
 index = rhs[j].find(l)  
 k = rhs[j].split(l)  
 s = k[0] + r + k[1]  
 rhs[j] = s  
 print(**"======================================================================================"**)  
 print(**"After copy propogation: "**)  
 **for** i **in** range(len(lhs)):  
 print(lhs[i] + **' : '** + rhs[i])  
  
  
**def** constantFolding(lhs, rhs):  
 print()  
 sz = len(lhs)  
 flag = 0  
 **for** i **in** range(sz):  
 l = lhs[i]  
 r = rhs[i]  
 **if** (**'+' in** rhs[i]):  
 k = rhs[i].split(**'+'**)  
 **if** k[0].isdigit() **and** k[1].isdigit():  
 s = int(k[0]) + int(k[1])  
 rhs[i] = str(s)  
 flag = 1  
  
 **elif** (**'-' in** rhs[i]):  
 k = rhs[i].split(**'-'**)  
 **if** k[0].isdigit() **and** k[1].isdigit():  
 s = int(k[0]) - int(k[1])  
 rhs[i] = str(s)  
 flag = 1  
  
 **elif** (**'\*' in** rhs[i]):  
 k = rhs[i].split(**'\*'**)  
 **if** k[0].isdigit() **and** k[1].isdigit():  
 s = int(k[0]) \* int(k[1])  
 rhs[i] = str(s)  
 flag = 1  
  
 **elif** (**'/' in** rhs[i]):  
 k = rhs[i].split(**'/'**)  
 **if** k[0].isdigit() **and** k[1].isdigit():  
 s = int(k[0]) / int(k[1])  
 rhs[i] = str(s)  
 flag = 1  
 print(**"======================================================================================"**)  
 print(**"After constant folding: "**)  
 **for** i **in** range(len(lhs)):  
 print(lhs[i] + **' : '** + rhs[i])  
  
  
**def** algebraicSimpliAndStrengthReducn(lhs, rhs):  
 print()  
 sz = len(lhs)  
 **for** i **in** range(sz):  
 **if** rhs[i].find(**'+'**) >= 0 **and** rhs[i].find(**'0'**) >= 0:  
 k = rhs[i].split(**'+'**)  
 **if** k[0] == **'0'**:  
 rhs[i] = k[1]  
 **else**:  
 rhs[i] = k[0]  
  
 **elif** rhs[i].find(**'-'**) >= 0 **and** rhs[i].find(**'0'**) >= 0:  
 k = rhs[i].split(**'-'**)  
 **if** k[0] == **'0'**:  
 rhs[i] = **'-'** + k[1]  
 **else**:  
 rhs[i] = k[0]  
  
 **elif** rhs[i].find(**'\*'**) >= 0 **and** rhs[i].find(**'1'**) >= 0:  
 k = rhs[i].split(**'\*'**)  
 **if** k[0] == **'1'**:  
 rhs[i] = k[1]  
 **else**:  
 rhs[i] = k[0]  
  
 **elif** rhs[i].find(**'/'**) >= 0 **and** rhs[i].find(**'1'**) >= 0:  
 k = rhs[i].split(**'/'**)  
 **if** k[1] == **'1'**:  
 rhs[i] = k[0]  
  
 **elif** rhs[i].find(**'\*'**) >= 0 **and** rhs[i].find(**'0'**) >= 0:  
 rhs[i] = **'0'  
  
 elif** rhs[i].find(**'^'**) >= 0 **and** rhs[i].find(**'2'**) >= 0:  
 k = rhs[i].split(**'^'**)  
 rhs[i] = k[0] + **'\*'** + k[0]  
 print(**"======================================================================================"**)  
 print(**"After algebraic simplification: "**)  
 **for** i **in** range(len(lhs)):  
 print(lhs[i] + **' : '** + rhs[i])  
  
  
**def** commonSubExp(lhs, rhs):  
 print()  
 sz = len(lhs)  
 **for** i **in** range(sz):  
 l = lhs[i]  
 r = rhs[i]  
 **for** j **in** range(i + 1, sz):  
 **if** r.find(lhs[j]) >= 0:  
 **break  
 else**:  
 **if** rhs[i] == rhs[j]:  
 rhs[j] = l  
 print(**"======================================================================================"**)  
 print(**"After Common subexpression ellimination: "**)  
 **for** i **in** range(len(lhs)):  
 print(lhs[i] + **' : '** + rhs[i])  
  
  
  
  
*# input = open('inp8.txt','r')*input\_list = []  
**with** open(**'input.txt'**) **as** f:  
 input\_list = f.readlines()  
  
print(input\_list)  
  
input\_list = [x.strip() **for** x **in** input\_list]  
print(input\_list)  
  
**for** i **in** range(len(input\_list)):  
 k = input\_list[i].split(**"="**)  
 lhs.append(k[0])  
 rhs.append(k[1])  
  
print(**"======================================================================================"**)  
**for** i **in** range(len(lhs)):  
 print(lhs[i] + **' : '** + rhs[i])  
  
*# lhs, rhs= deadCode(lhs, rhs)  
  
# print("After dead code elimination: ")  
# for i in range(len(lhs)):  
# print(lhs[i]+ ' : '+rhs[i])  
  
# copyProp(lhs, rhs)  
  
# constantFolding(lhs, rhs)  
  
# constantProp(lhs, rhs)  
  
# algebraicSimpliAndStrengthReducn(lhs, rhs)  
  
# commonSubExp(lhs, rhs)*algebraicSimpliAndStrengthReducn(lhs, rhs)  
constantProp(lhs, rhs)  
copyProp(lhs, rhs)  
constantFolding(lhs, rhs)  
constantProp(lhs, rhs)  
commonSubExp(lhs, rhs)  
copyProp(lhs, rhs)  
lhs, rhs = deadCode(lhs, rhs)  
print(**"======================================================================================"**)

**OUTPUT :**











