Om Shete
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
TE C32
2103163

Experiment No : 8

Aim : Write a program to implement Code Generation Optimization.

Theory :

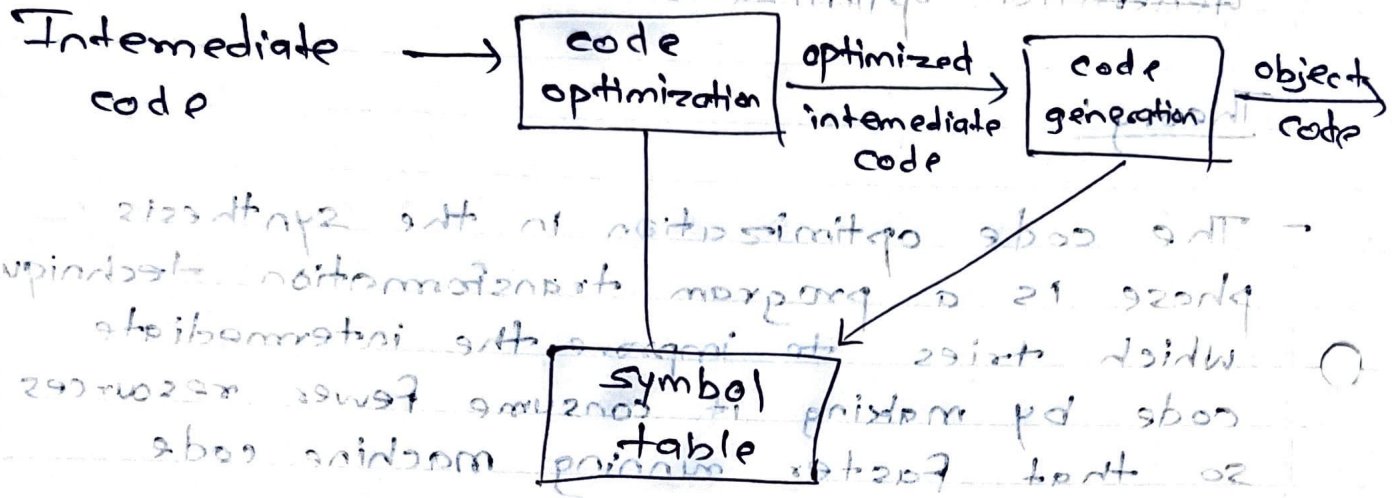
- The code optimization in the synthesis phase is a program transformation technique, which tries to improve the intermediate code by making it consume fewer resources so that faster running machine code will result.
- Compiler optimizing process should meet the following objective :
  1) The optimization must be correct, it must not, in any way, change the meaning of the program.
  2) Optimization should increase the speed and the performance of program.
- Types of Code Optimization :-

1) Machine Independent Optimization

⇒ This code optimization phase attempts to improve the optimization phase at intemediate code to get better target code as output.

2) Machine Dependent Optimization

⇒ Machine Dependent optimization is done after the target code has been generated and when the code is transformed according to


Intemediate code
code optimization
optimized intemediate code
code generation
object code
symbol table

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

target machine architecture.

- Ways to optimize code :-

i) Compile time evaluation

⇒

A = 2 * (22.0 / 7.0) * r

Perform 2 * (22.0 / 7.0) * r at compiler time.

ii) Variable propagation

⇒

```
c = a * b          |          c = a * b
x = a              |          x = a
till               | optimized  till
d = x * b + 4      |          d = a * b + 4
```

iii) Constant propagation

⇒ If variable is a constant, then replace variable constant.

- Conclusion: In this experiment, I learned about the code optimization and its technique.

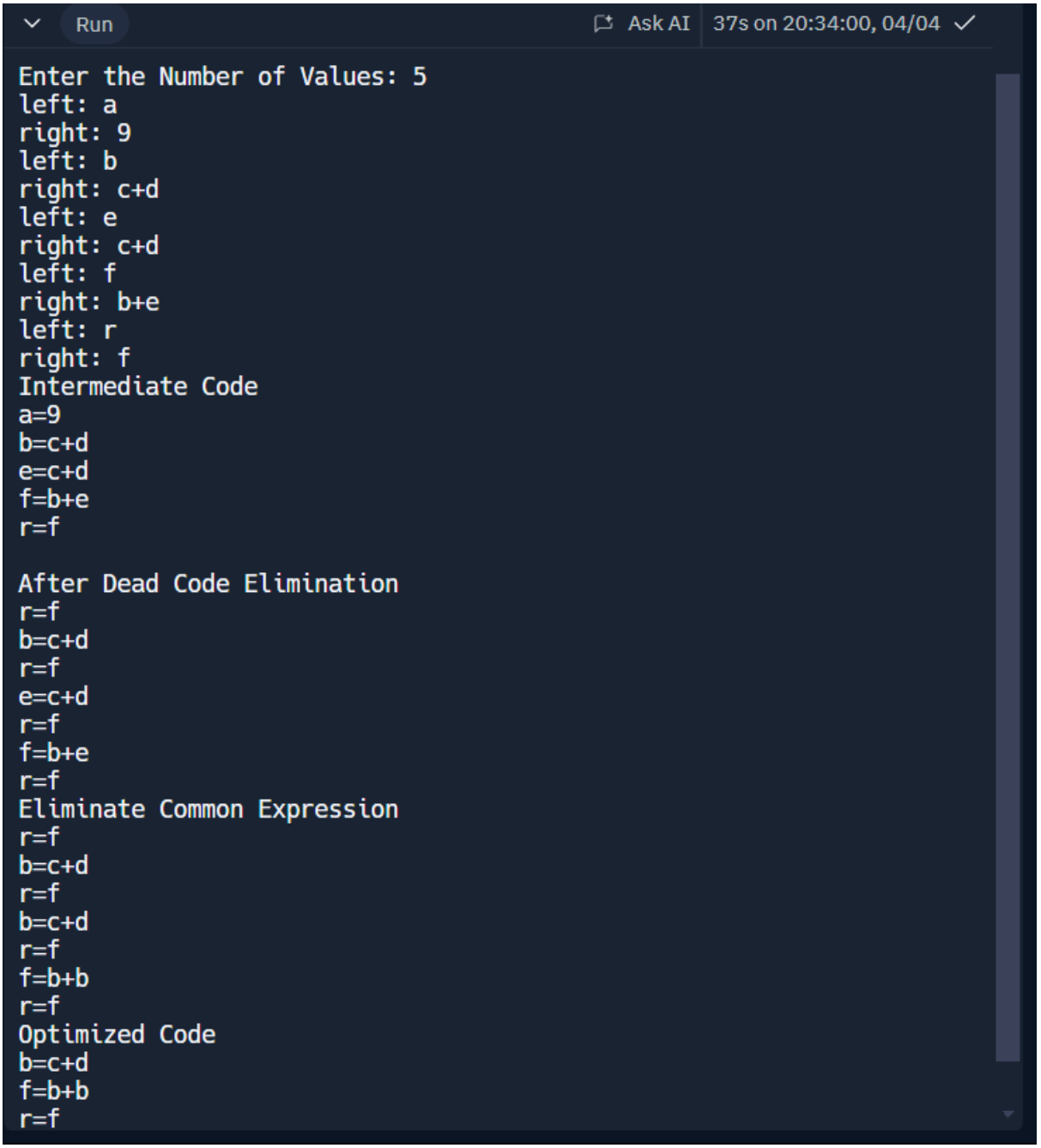
28/03/2024

Code:

```
class OP:

  def __init__(self, l, r):
    self.l = l
    self.r = r


op = []
pr = []


def main():
  n = int(input("Enter the Number of Values: "))
  for i in range(n):
    l = input("left: ")
    r = input("right: ")
    op.append(OP(l, r))

  print("Intermediate Code")
  for item in op:
    print(item.l + "=" + item.r)

  z = 0
  for i in range(n - 1):
    temp = op[i].l
    for j in range(n):
      p = op[j].r.find(temp)
      if p != -1:
        pr.append(OP(op[i].l, op[i].r))
        z += 1
    pr.append(OP(op[n - 1].l, op[n - 1].r))
    z += 1

  print("\nAfter Dead Code Elimination")
  for item in pr:
    print(item.l + "=" + item.r)
```

```
  for m in range(z):
    tem = pr[m].r
    for j in range(m + 1, z):
      p = tem.find(pr[j].r)
      if p != -1:
        t = pr[j].l
        pr[j].l = pr[m].l
        for i in range(z):
          l = pr[i].r.find(t)
          if l != -1:
            pr[i].r = pr[i].r[:l] + pr[m].l + pr[i].r[l + 1:]

  print("Eliminate Common Expression")
  for item in pr:
    print(item.l + "=" + item.r)

  for i in range(z):
    for j in range(i + 1, z):
      if pr[i].l == pr[j].l and pr[i].r == pr[j].r:
        pr[i].l = '\0'

  print("Optimized Code")
  for item in pr:
    if item.l != '\0':
      print(item.l + "=" + item.r)


if __name__ == "__main__":
  main()
```

Output:


Run
Ask AI
37s on 20:34:00, 04/04
Enter the Number of Values: 5
left: a
right: 9
left: b
right: c+d
left: e
right: c+d
left: f
right: b+e
left: r
right: f
Intermediate Code
a=9
b=c+d
e=c+d
f=b+e
r=f
After Dead Code Elimination
r=f
b=c+d
r=f
e=c+d
r=f
f=b+e
r=f
Eliminate Common Expression
r=f
b=c+d
r=f
b=c+d
r=f
f=b+b
r=f
Optimized Code
b=c+d
f=b+b
r=f