In [4]:

```

# Calculating expectation , assuming that he can attempt tasks in any order and r
def expectedReturn(policy,attempts):
    if len(policy) > 0 and attempts <= 10 :  # Max number of attempts is 10
        # we take the next task and increase attempts
        task = policy[0]

        second_attempt_pass = task[1] + expectedReturn(policy[1:], attempts+2)
        second_attempt_fail = 0 + expectedReturn(policy[1:], attempts+2)

        first_attempt_pass  =      task[2]     * ( task[1] +  expectedReturn(pol
        first_attempt_fail  =  (1-task[2])     * (    0    +  (task[2]/2)*second_
        return first_attempt_pass + first_attempt_fail
    else:
        return 0

#Task 2.2
# Find expectations of Policy A and B
tasks = [(1, 12, 0.25), (2, 4, 0.4), (3, 10, 0.35), (4, 5, 0.6), (5, 7, 0.45), (

import itertools
maxer = 0
for policy in list(itertools.permutations(tasks)) :
    #print("Current Policy :",policy)
    er = expectedReturn(policy,0)
    if er > maxer :
        maxer = er
        policyC = policy
    #print("Expected return of current policy :",er )


print("Best policy: ",policyC)
print("Expected Return of best policy: ",maxer)

# Explanation
# The function expectedReturn models the expectation of this scenario. A student
#( following this policy )
# Probability of passing : Expectation greater than 50%

```

```
Best policy:  ((3, 10, 0.35), (1, 12, 0.25), (6, 3, 0.5), (7, 50, 0.15), (4, 5,
0.6), (5, 7, 0.45), (2, 4, 0.4))
Expected Return of best policy:  30.512462500000005
```

In [ ]: