THỰC HÀNH MÔN TOÁN RỜI RẠC

Thời gian	:	3 buổi
Ngôn ngữ lập trình	:	**Python**, Java, C++
Phần mềm	:	**Visual Studio Code**, Eclipse, CodeBlocks
Yêu cầu	:	SV nắm bài trên lớp, có ý tưởng / sơ đồ khối của các thuật toán
Đánh giá	:	10% ĐQT.

1 Nguyên lý đếm cơ bản

1.1 Quy tắc cộng, nhân

```
m, n = 4, 3
counter = 0
for i in range(1, m+1):
    for j in range(1, n+1):
        counter += 1
        print(counter, i, j)
```

Mã 1: Ví dụ 1.7

```
n = 4
counter = 0
for i in range(1, n+1):
    for j in range(1, i+1):
        counter += 1

print(counter)
```

Mã 2: Ví dụ 1.8

```
from sympy import *

n, i = symbols('n i')
Sum(i, (i, 1, n)).doit().simplify()
```

Mã 3: Ví dụ 1.8

```
A = [[ 2,  0, -1],
     [ 1,  3, -2]]
B = [[ 0, -1,  1, 0],
     [ 2,  3, -1, 4],
     [-3,  0, -2, 1]]
```

Mã 4: Ví dụ 1.9

```
def matrix_mul(A, B):
    m, n = len(A), len(B)
    p = len(B[0])
```

```
    C = [[0 for j in range(p)] for i in range(m)]
    for i in range(m):
        for j in range(p):
            for k in range(n):
                C[i][j] += A[i][k] * B[k][j]
    return C

matrix_mul(A, B)
```

Mã 5: Ví dụ 1.9

```
import numpy as np

np.dot(A, B)
```

Mã 6: Ví dụ 1.9

```
def BubbleSort(x):
    n = len(x)
    for i in range(n-1):
        for j in range(n-1,i,-1):
            if x[j] < x[j-1]:
                x[j-1], x[j] = x[j], x[j-1]
    return x

BubbleSort([7, 9, 2, 5, 8])
```

Mã 7: Ví dụ 1.10

1.2 Hoán vị, chỉnh hợp

```
def factorial(n):
    p = 1
    for i in range(1, n+1):
        p *= i
    return p

factorial(8)
```

Mã 8: Ví dụ 1.13

```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n-1)

factorial(8)
```

Mã 9: Ví dụ 1.13

```
from sympy import *

factorial(8)
```

Mã 10: Ví dụ 1.13

```
def permutations(a):
    n = len(a)
    if n == 1:
        return [a]
    P = []
    for i in range(n):
        b = a.copy()
        x = b.pop(i)
        for p in permutations(b):
            p = [x] + p
            P.append(p)
    return P
```

Mã 11: Ví dụ 1.14

```
import itertools

list( itertools.permutations([1, 2, 3]) )
```

Mã 12: Ví dụ 1.14

```
def binary_strs(n):
    if n==1:
        return ['0', '1']
    A = []
    for s in binary_strs(n-1):
        A.append('0' + s)
    for s in binary_strs(n-1):
        A.append('1' + s)
    return A

binary_strs(3)
```

Mã 13: Ví dụ 1.15

```
import itertools
list(itertools.product([0, 1], repeat=3))
```

Mã 14: Ví dụ 1.15

```
def P(n, r):
    p = 1
    for i in range(r):
```

```
        p *= n - i
    return p

P(8, 5)
```

Mã 15: Ví dụ 1.16

```
from sympy import *

n, r = 8, 5
factorial(n) / factorial(n-r)
```

Mã 16: Ví dụ 1.16

```
def permutations(a, r):
    if r == 1:
        return [[i] for i in a]
    P = []
    n = len(a)
    for i in range(n):
        b = a.copy()
        x = b.pop(i)
        for p in permutations(b, r-1):
            p = [x] + p
            P.append(p)
    return P

permutations([1, 2, 3, 4], 3)
```

Mã 17: Ví dụ 1.17

```
import itertools

list( itertools.permutations([1, 2, 3, 4], 3) )
```

Mã 18: Ví dụ 1.17

1.3 Tổ hợp

```
from sympy import *

binomial(10, 4)
```

Mã 19: Ví dụ 1.18

```
def binomial(n, r):
    p = 1
    for i in range(r):
        p = p * (n-i) // (i+1)
```

```
    return p

binomial(10, 4)
```

Mã 20: Ví dụ 1.18

```
def combinations(a, r):
    if r == 1:
        return [[i] for i in a]
    n = len(a)
    if r == n:
        return [a]
    C = []
    for c in combinations(a[1:], r-1):
        c = [a[0]] + c
        C.append(c)
    for c in combinations(a[1:], r):
        C.append(c)
    return C

combinations([1, 2, 3, 4, 5], 3)
```

Mã 21: Ví dụ 1.19

```
import itertools

list( itertools.combinations([1, 2, 3, 4, 5], 3) )
```

Mã 22: Ví dụ 1.19

```
from sympy import *

x, y = symbols('x y')
( (x + y)**2 ).expand()
```

Mã 23: Ví dụ 1.21

```
def binomial(n, r):
    if r == 0 or r == n:
        return 1
    return binomial(n-1, r-1) + binomial(n-1, r)

binomial(10, 4)
```

Mã 24: Định lý 1.2

```
def binomial(n, r):
    a = [1]
    for i in range(1, n+1):
```

```
        for j in range(i-1, 0, -1):
            a[j] += a[j-1]
        a.append(1)
    return a[r]

binomial(10, 4)
```

Mã 25: Định lý 1.2

1.4 Hoán vị lặp

```
def permutations_with_replacement(a, n):
    r = len(a)
    if sum(n) == 0:
        return [[]]
    P = []
    for i in range(r):
        if n[i] > 0:
            n_ = n.copy()
            n_[i] -= 1
            for p in permutations_with_replacement(a, n_):
                p = [a[i]] + p
                P.append(p)
    return P

permutations_with_replacement(['A', 'B', 'L'], [1, 1, 2])
```

Mã 26: Ví dụ 1.23

```
def walks(a, b, x, y):
    if a == x:
        return ['U' * (y-b)]
    if b == y:
        return ['R' * (x-a)]
    W = []
    for w in walks(a+1, b, x, y):
        w = 'R' + w
        W.append(w)
    for w in walks(a, b+1, x, y):
        w = 'U' + w
        W.append(w)
    return W
```

Mã 27: Ví dụ 1.24

```
from sympy import *
```

```
x, y, z = symbols('x y z')
( (x + y + z)**7 ).expand()

( (x + y + z)**7 ).expand().coeff(x * y**5 * z)

a, b, c = symbols('a b c')
expr = (a - 2*b + 3*c + 5)**10
expr.expand().coeff(a**4 * b * c**3)
```

Mã 28: Ví dụ 1.25

1.5 Tổ hợp lặp

```
import itertools

list( itertools.combinations_with_replacement(['A', 'B', 'C'], 4) )
```

Mã 29: Ví dụ 1.26

```
def combinations_with_replacement(a, r):
    n = len(a)
    if n == 1:
        return [a * r]
    if r == 1:
        return [[i] for i in a]
    C = []
    for c in combinations_with_replacement(a, r-1):
        c = [a[0]] + c
        C.append(c)
    for c in combinations_with_replacement(a[1:], r):
        C.append(c)
    return C

combinations_with_replacement(['A', 'B', 'C'], 4)
```

Mã 30: Ví dụ 1.26

```
from sympy import *

r = symbols('r')
Sum(binomial(6+r-1, r), (r, 0, 9)).doit()
```

Mã 31: Ví dụ 1.29

```
def summands(n):
    if n == 1:
        return [[1]]
    S = []
```

```
    for i in range(1, n):
        for s in summands(n-i):
            s = [i] + s
            S.append(s)
    S.append([n])
    return S

summands(3)
```

Mã 32: Ví dụ 1.30

1.6 Sinh các hoán vị và tổ hợp

```
def compare(a, b):
    m, n = len(a), len(b)
    i = 0
    while i < m and i < n and a[i] == b[i]:
        i += 1
    if i == m == n:
        return('=')
    if i == m < n:
        return('<')
    if i == n < m:
        return('>')
    if i < m and i < n:
        if a[i] < b[i]:
            return('<')
        else:
            return('>')

compare([4, 1, 2], [4, 1, 2, 3])

compare([3, 1, 4], [3, 1, 2, 5])
```

Mã 33: Ví dụ 1.31

```
def next_permutations(a):
    n = len(a)
    k = n - 1
    while k >= 1 and a[k-1] > a[k]:
        k -= 1

    if k == 0:
        return None

    i = n - 1
```

```
    while a[i] < a[k-1]:
        i -= 1
    a[k-1], a[i] = a[i], a[k-1]

    b = a[k:]
    b.reverse()
    return a[:k] + b

next_permutations([3, 6, 2, 5, 4, 1])
```

Mã 34: Ví dụ 1.32

```
def next_combinations(n, a):
    r = len(a)
    i = r - 1
    while i >= 0 and a[i] == n - r + (i + 1):
        i -= 1

    if i == -1:
        return None

    return a[:i] + [a[i] + j for j in range(1, r-i+1)]

next_combinations(6, [1, 2, 5, 6])
```

Mã 35: Ví dụ 1.33

```
def next_bin_str(a):
    n = len(a)
    i = n - 1
    while i >=0 and a[i] == 1:
        i -= 1

    if i == -1:
        return None

    for j in range(i, n):
        a[j] = 1 - a[j]
    return a

a = [1, 0, 0, 0, 1, 0, 0, 1, 1, 1]
next_bin_str(a)
```

Mã 36: Ví dụ 1.34

1.7 Số Catalan

```
def catalan_walks(a, b, n):
    if a == n:
        return ['U' * (n-b)]
    W = []
    if a == b:
        for w in catalan_walks(a+1, b, n):
            w = 'R' + w
            W.append(w)
    if a > b:
        for w in catalan_walks(a+1, b, n):
            w = 'R' + w
            W.append(w)
        for w in catalan_walks(a, b+1, n):
            w = 'U' + w
            W.append(w)
    return W

catalan_walks(0, 0, 3)
```

Mã 37: Ví dụ 1.35

```
from sympy import *

[binomial(2*n, n) / (n+1) for n in range(11)]
```

Mã 38: Ví dụ 1.35