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.

9 Hệ thức đệ quy

9.1 Python

```python
from sympy import *

n = symbols('n')
F = symbols('F', cls=Function)
rsolve(
    -F(n) + F(n-1) + F(n-2),
    F(n),
    {F(0): 0, F(1): 1}
)
```

Mã 66: Ví dụ 9.2

9.2 Hệ thức đệ quy tuyến tính cấp 1

```python
a = 100
for _ in range(3):
    a = a + a * 5 / 100
    print(a)
```

Mã 67: Ví dụ 9.3

```python
from sympy import *

n = symbols('n')
a = symbols('a', cls=Function)

sol = rsolve(
    a(n) - 1.05*a(n-1),
    a(n),
    {a(0): 100}
)
sol
```

Mã 68: Ví dụ 9.3

```
from sympy import *

n, P, r, S = symbols('n P r S')
a = symbols('a', cls=Function)
sol = rsolve(
    -a(n) + (1+r) * a(n-1) - P,
    a(n),
    {a(0): S}
)
sol

sol.simplify()

sol_P = solve(sol, P)
sol_P

sol_P[0]

sol_P[0].subs({S: 100, r: 0.01, n: 12})
```

Mã 69: Ví dụ 9.4

```
def summands(n):
    if n == 1:
        return [[1]]
    S = []
    for s in summands(n-1):
        s[0] += 1
        S.append(s)
    for s in summands(n-1):
        s = [1] + s
        S.append(s)
    return S

summands(3)
```

Mã 70: Ví dụ 9.5

```
def BubleSort(x):
    n = len(x)
    if n == 1:
        return x
    for i in range(n-1, 0, -1):
        if x[i] < x[i-1]:
            x[i], x[i-1] = x[i-1], x[i]
    return [x[0]] + BubleSort(x[1:])
```

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

Mã 71: Ví dụ 9.6

```
def hanoi_tower(n, A, B, C):
    if n == 1:
        return [[1, A, B]]
    return hanoi_tower(n-1, A, C, B) + [[n, A, B]] + hanoi_tower(n-1, C,
    B, A)

hanoi_tower(3, 'A', 'B', 'C')
```

Mã 72: Ví dụ 9.8

```
def quaternary_strs(n):
    if n == 1:
        return [[i] for i in range(4)]
    S = []
    for i in range(4):
        for s in quaternary_strs(n-1):
            s = [i] + s
            S.append(s)
    return S

quaternary_strs(2)

def quaternary_strs_1s_even(n):
    if n == 1:
        return [[0], [2], [3]]
    S = []
    for s in quaternary_strs(n-1):
        if s not in quaternary_strs_1s_even(n-1):
            s = [1] + s
            S.append(s)
    for i in [0, 2, 3]:
        for s in quaternary_strs_1s_even(n-1):
            s = [i] + s
            S.append(s)
    return S

quaternary_strs_1s_even(2)
```

Mã 73: Ví dụ 9.9

9.3 Hệ thức đệ quy tuyến tính thuần nhất hệ số hằng

```
from sympy import *

n = symbols('n')
a = symbols('a', cls=Function)
rsolve(
    -a(n) + 3*a(n-2) - 2*a(n-3),
    a(n)
)

rsolve(
    -a(n) + 3*a(n-2) - 2*a(n-3),
    a(n),
    {a(0): 5, a(1): -1, a(2): 2}
)
```

Mã 74: Ví dụ 9.12

```
from sympy import *

x = symbols('x')
P = x**3 - 3*x + 2
P.factor()

C1, C2, C3 = symbols('C1 C2 C3')
ans = C1 + C2*n + (-2)**n * C3

[ans.subs(n, i) for i in [0, 1, 2]]

eqns = [ans.subs(n, i) - a for i, a in zip([0, 1, 2], [5, -1, 2])]
solve(eqns)
```

Mã 75: Ví dụ 9.12

```
from sympy import *

n = symbols('n', integer=True)
a = symbols('a', cls=Function)
ans = rsolve(
    a(n+1) - 2*a(n) + 2*a(n-1),
    a(n),
    {a(0): 1, a(1): 2}
)
ans

polar = lambda z: abs(z) * E**(I*arg(z))
```

```
print(ans)

ans = (Rational(1, 2) + I/2) * polar(1 - I)**n + (Rational(1, 2) - I/2)
   * polar(1 + I)**n
ans

re(ans)
```

Mã 76: Ví dụ 9.13

```
from sympy import *

x = symbols('x')
P = x**2 - 2*x + 2
solve(P)

x = 1 + I
r = abs(x)
phi = arg(x)

C1, C2 = symbols('C1 C2')
ans = r**n * (C1 * cos(n*phi) + C2 * sin(n*phi))

[ans.subs(n, i).simplify() for i in [0, 1]]

solve([ans.subs(n, i).simplify() - a for i, a in zip([0, 1], [1, 2])])
```

Mã 77: Ví dụ 9.13

```
def subsets_with_condition(n):
    if n == 1:
        return [[], [1]]
    if n == 2:
        return [[], [1], [2]]
    S = []
    for s in subsets_with_condition(n-1):
        S.append(s)
    for s in subsets_with_condition(n-2):
        s.append(n)
        S.append(s)
    return S

subsets_with_condition(3)
```

Mã 78: Ví dụ 9.14

```
def binary_strs(n):
```

```
    if n == 1:
        return ['0', '1']
    if n == 2:
        return ['01', '10', '11']
    S = []
    for s in binary_strs(n-2):
        s = '01' + s
        S.append(s)
    for s in binary_strs(n-1):
        s = '1' + s
        S.append(s)
    return S

binary_strs(3)
```

Mã 79: Ví dụ 9.16

```
from sympy import *

n = symbols('n')
a = symbols('a', cls=Function)
ans = rsolve(
    a(n) - 2*a(n-2),
    a(n),
    {a(1): 1, a(2): 2}
)
ans

k = symbols('k', integer=True)

ans.subs(n, 2*k).simplify()
ans.subs(n, 2*k+1).simplify()
```

Mã 80: Ví dụ 9.18

```
def symmetric_summands(n):
    if n == 1:
        return [[1]]
    if n == 2:
        return [[2], [1, 1]]
    S = []
    for s in symmetric_summands(n-2):
        s[0] += 1
        s[-1] += 1
        S.append(s)
    for s in symmetric_summands(n-2):
```

```
        s = [1] + s + [1]
        S.append(s)
    return S

symmetric_summands(4)
```

Mã 81: Ví dụ 9.18