monolithicFibonacci.c

```
int count = 0;
int next = 0;
int first = 0;
int second = 1;
int i = 2;

scanf("%d", &count);

if (count >= 1)
    printf("\n%d\n", first);
else
    goto end;

if (count >= 2)
    printf("%d\n", second);
else
    goto end;

redo: if(count > i){
    next = first + second;
    printf("%d\n", next);
    first = second;
    second = next;
    i++;
    goto redo;
}

    end: return 0;
```

monolithicFibonacci.c => Monolithic Algorithm

R1: Do C = 0 goto R2; = A
R2: Do N = 0 goto R3; = B
R3: Do F = 0 goto R4; = C
R4: Do S = 1 goto R5; = D
R5: Do I = 2 goto R6; = E
R6: Do Read(C) goto R7; = F
R7: If (C >= 1) goto R8 else goto Rx; = G else Rx
R8: Do Write(F) goto R9; = H
R9: If (C >= 2) goto R10 else goto Rx; = J else Rx
R10: Do Write(S) goto R11; = K
R11: If (C > I) goto R12 else goto Rx; = L else Rx
R12: Do N = F + S goto R13; = M
R13: Do Write(N) goto R14; = M
R14: Do F = S goto R15; = M
R15: Do S = N goto R16; = M
R16: Do I = I + 1 goto R11; = M

Monolithic Algorithm => Trace Machine

n = 4

(1, A)
(2, AB)
(3, ABC)
(4, ABCD)
(5, ABCDE)
(6, ABCDEF)
(7, ABCDEFG)
(8, ABCDEFGH)
(9, ABCDEFGHJ)

(10, ABCDEFGHJK)
(11, ABCDEFGHJKL)
(12, ABCDEFGHJKLM)
(11, ABCDEFGHJKLML)
(12, ABCDEFGHJKLMLM)
(11, ABCDEFGHJKLMLML)
(Rx, ABCDEFGHJKLMLML)

iterativeFibonacci.c

```
int count = 0;
int next = 0;
int first = 0;
int second = 1;
int i = 2;

scanf("%d", &count);

if (count >= 1)
    printf("\n%d\n", first);

if (count >= 2)
    printf("%d\n", second);

while(count > i){
    next = first + second;
    printf("%d\n", next);
    first = second;
    second = next;
    i++;
}
```

```
return 0;
```

iterativeFibonacci.c => Monolithic Algorithm

R1: Do C = 0 goto R2; = A
R2: Do N = 0 goto R3; = B
R3: Do F = 0 goto R4; = C
R4: Do S = 1 goto R5; = D
R5: Do I = 2 goto R6; = E
R6: Do Read(C) goto R7; = F
R7: If (C >= 1) goto R8 else goto Rx; = G else Rx
R8: Do Write(F) goto R9; = H
R9: If (C >= 2) goto R10 else goto Rx; = J else Rx
R10: Do Write(S) goto R11; = K
R11: If (C > I) goto R12 else goto Rx; = L else Rx
R12: Do N = F + S goto R13; = M
R13: Do Write(N) goto R14; = M
R14: Do F = S goto R15; = M
R15: Do S = N goto R16; = M
R16: Do I = I + 1 goto R11; = M

Monolithic Algorithm => Trace Machine

n = 4

(1, A)
(2, AB)
(3, ABC)
(4, ABCD)
(5, ABCDE)
(6, ABCDEF)
(7, ABCDEFG)

(8, ABCDEFGH)
(9, ABCDEFGHJ)
(10, ABCDEFGHJK)
(11, ABCDEFGHJKL)
(12, ABCDEFGHJKLM)
(11, ABCDEFGHJKLML)
(12, ABCDEFGHJKLMLM)
(11, ABCDEFGHJKLMLML)
(Rx, ABCDEFGHJKLMLML)

recursiveFibonacci.c

```
int fibonacci(int n)    F
{
    if (n <= 1)    G
        return n;     H
    else
        return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
    int n = 0;    A
    int i = 0;    B

    scanf("%d", &n);    C

    while (n > i)    D
    {
        printf("%d\n", fibonacci(i));    F
        i++;    K
    }
```

```
    return 0;
}
```

recursiveFibonacci.c => Trace Machine

n = 4

(1, A)
(2, AB)
(3, ABC)
(4, ABCD) *i = 0*
    (5, ABCDF)
    (6, ABCDFG)
    (7, ABCDFGH)
(8, ABCDFGHK)
(4, ABCDFGHKD) *i = 1*
    (5, ABCDFGHKDF)
    (6, ABCDFGHKDFG)
    (7, ABCDFGHKDFGH)
(8, ABCDFGHKDFGHK)
(4, ABCDFGHKDFGHKD) *i = 2*
    (5, ABCDFGHKDFGHKDF)
    (6, ABCDFGHKDFGHKDFG)
    (5, ABCDFGHKDFGHKDFGF)
    (6, ABCDFGHKDFGHKDFGFG)
    (7, ABCDFGHKDFGHKDFGFGH)
    (5, ABCDFGHKDFGHKDFGFGHF)
    (6, ABCDFGHKDFGHKDFGFGHFG)
    (7, ABCDFGHKDFGHKDFGFGHFGH)
(8, ABCDFGHKDFGHKDFGFGHFGHK)
(4, ABCDFGHKDFGHKDFGFGHFGHKD) *i = 3*
    (5, ABCDFGHKDFGHKDFGFGHFGHKDF)
    (6, ABCDFGHKDFGHKDFGFGHFGHKDFG)

(5, ABCDFGHKDFGHKDFGFGHFGHKDFGF)
(6, ABCDFGHKDFGHKDFGFGHFGHKDFGFG)
(5, ABCDFGHKDFGHKDFGFGHFGHKDFGFGF)
(6, ABCDFGHKDFGHKDFGFGHFGHKDFGFGFG)
(7, ABCDFGHKDFGHKDFGFGHFGHKDFGFGFGH)
(5, ABCDFGHKDFGHKDFGFGHFGHKDFGFGFGHF)
(6, ABCDFGHKDFGHKDFGFGHFGHKDFGFGFGHFG)
(7, ABCDFGHKDFGHKDFGFGHFGHKDFGFGFGHFGH)
(5, ABCDFGHKDFGHKDFGFGHFGHKDFGFGFGHFGHF)
(6, ABCDFGHKDFGHKDFGFGHFGHKDFGFGFGHFGHFG)
(7, ABCDFGHKDFGHKDFGFGHFGHKDFGFGFGHFGHFGH)
(8, ABCDFGHKDFGHKDFGFGHFGHKDFGFGFGHFGHFGHK)
(5, ABCDFGHKDFGHKDFGFGHFGHKDFGFGFGHFGHFGHKD)
(Rx, ABCDFGHKDFGHKDFGFGHFGHKDFGFGFGHFGHFGHKD)

Comparison of Tracing Machines

M (Rx, ABCDEFGHJKLMLML)

I (Rx, ABCDEFGHJKLMLML)

R (Rx,
ABCDFGHKDFGHKDFGFGHFGHKDFGFGFGHFGHFGHKD)

Conclusion

M = I

M ≠ R

I ≠ R