Lab 8.1 #8

Tracing:

int f(int n) {

if(n < 2) // 2

return 0; // 3

else

return 1 + f(n/2); // 4

}

|  |  |  |
| --- | --- | --- |
| Statements being executed | ACTION | Current function call |
| 1 | Call func f with arg 12 | f(12) |
| 2,4 | Call f with 12 / 2 = 6 | F(12) |
| 2,4 | Call f with 6/2 = 3 | F(6) |
| 2,4 | Call f with 3/2 = 1 | F(3) |
| 2,3 | Return 0 | F(1) |
| 4 | Return 1 + 0 = 1 | F(3) |
| 4 | Return 1 + 1 = 2 | F(6) |
| 4 | Return 2 + 1 = 3 | F(12) |
| 1 | Assign 3 to x | Originally calling function |

X = f(12); // 1

Tracing my code:

int recFibonacci(int n) {

if (n == 1 || n == 2) // 2

return 1; // 3

else

return recFibonacci(n - 1) + recFibonacci(n - 2); // 4

}

X = recFibonacci(6); // 1

Using rf = recFibonacci for convenience.

|  |  |  |
| --- | --- | --- |
| Statements being executed | ACTION | Current function call |
| 1 | Call rF(6) | Main Driver |
| 2, 4 | Call rF(5) | rF(6) |
| 2, 4 | Call rF(4) | rF(5) |
| 2, 4 | Call rF(3) | rF(4) |
| 2, 4 | Call rF(2) | rF(3) |
| 2, 3 | Return 1 | rF(2) |
| 4 | Call rF(1) | rF(3) |
| 2, 3 | Return 1 | rF(1) |
| 4 | Return 1 + 1 = 2 | rF(3) |
| 4 | Call rF(2) | rF(4) |
| 2, 3 | Return 1 | rF(2) |
| 4 | Return 1 + 2 = 3 | rF(4) |
| 4 | Call rF(3) | rF(5) |
| 2, 4 | Call rF(2) | rF(3) |
| 2, 3 | Return 1 | rF(2) |
| 4 | Call rF(1) | rF(3) |
| 2, 3 | Return 1 | rF(1) |
| 4 | Return 1 + 1 = 2 | rF(3) |
| 4 | Return 2 + 3 = 5 | rF(5) |
| 4 | Call rF(4) | rF(6) |
| 2, 4 | Call rF(3) | rF(4) |
| 2, 4 | Call rF(2) | rF(3) |
| 2, 3 | Return 1 | rF(2) |
| 4 | Call rF(1) | rF(3) |
| 2, 3 | Return 1 | rF(1) |
| 4 | Return 1 + 1 = 2 | rF(3) |
| 4 | Call rF(2) | rF(4) |
| 2, 3 | Return 1 | rF(2) |
| 4 | Return 1 + 2 = 3 | rF(4) |
| 4 | Return 3 + 5 = 8 | rF(6) |
| 1 | Assign 8 to x | Main Driver |

Tracing Recursive Functions – Version 2

|  |  |  |
| --- | --- | --- |
| Function Call | Run-Time Stack | Action |
|  | Empty |  |
| F(12) | {12, ?, 1} | Push AR |
| F(6) | {12, ?, 1}, {6, ?, 4} | Push AR |
| F(3) | {12, ?, 1}, {6, ?, 4}, {3, ?, 4} | Push AR |
| F(1) | {12, ?, 1}, {6, ?, 4}, {3, ?, 4}, {1, ?, 4} | Push AR |
| F(1) | {12, ?, 1}, {6, ?, 4}, {3, ?, 4}, {1, 0, 4} | Return 0 |
| F(3) | {12, ?, 1}, {6, ?, 4}, {3, 1, 4} | Return 1 |
| F(6) | {12, ?, 1}, {6, 2, 4} | Return 2 |
| F(12) | {12, 3, 1} | Return 3 |
| Main driver | {Whatever was in the main driver} | Assign 3 to x |

Tracing my code:

int recFibonacci(int n) {

if (n == 1 || n == 2) // 2

return 1; // 3

else

return recFibonacci(n - 1) + recFibonacci(n - 2); // 4

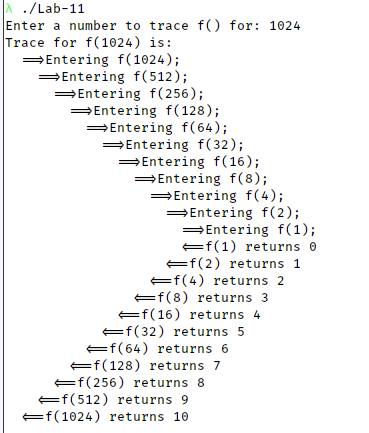
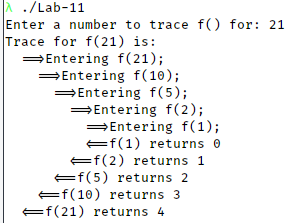
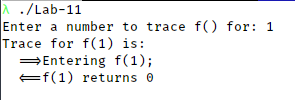
}

X = recFibonacci(6); // 1

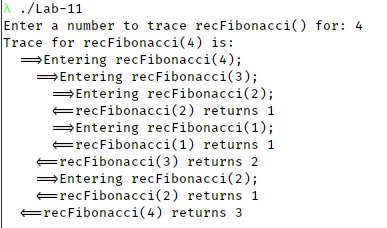
Using rf = recFibonacci for convenience.

|  |  |  |
| --- | --- | --- |
| Function Call | Stack | Action |
| Main Driver | {Whatever driver had} | Push AR |
| rF(6) | {6, ?, 1} | Push AR |
| rF(5) | {6,?,1},{5,?,4} | Push AR |
| rF(4) | {6,?,1},{5,?,4},{4,?,4} | Push AR |
| rF(3) | {6,?,1},{5,?,4},{4,?,4},{3,?,4} | Push AR |
| rF(2) | {6,?,1},{5,?,4},{4,?,4},{3,?,4}, {2,?,4} | Push AR |
| rF(2) | {6,?,1},{5,?,4},{4,?,4},{3,?,4}, {2,1,4} | Return 1 |
| rF(3) | {6,?,1},{5,?,4},{4,?,4},{3,?,4} | Call rF(1) |
| rF(1) | {6,?,1},{5,?,4},{4,?,4},{3,?,4},{1,?,4} | Push AR |
| rF(1) | {6,?,1},{5,?,4},{4,?,4},{3,?,4},{1,1,4} | Return 1 |
| rF(3) | {6,?,1},{5,?,4},{4,?,4},{3,2,4} | Return 1 + 1 = 2 |
| rF(4) | {6,?,1},{5,?,4},{4,?,4} | Call rF(2) |
| rF(2) | {6,?,1},{5,?,4},{4,?,4},{2,?,4} | Push AR |
| rF(2) | {6,?,1},{5,?,4},{4,?,4},{2,1,4} | Return 1 |
| rF(4) | {6,?,1},{5,?,4},{4,3,4} | Return 1 + 2 = 3 |
| rF(5) | {6,?,1},{5,?,4} | Call rF(3) |
| rF(3) | {6,?,1},{5,?,4},{3,?,4} | Push AR |
| rF(2) | {6,?,1},{5,?,4},{3,?,4},{2,?,4} | Push AR |
| rF(2) | {6,?,1},{5,?,4},{3,?,4},{2,1,4} | Return 1 |
| rF(3) | {6,?,1},{5,?,4},{3,?,4}, | Call rF(1) |
| rF(1) | {6,?,1},{5,?,4},{3,?,4},{1,?,4} | Push AR |
| rF(1) | {6,?,1},{5,?,4},{3,?,4},{1,1,4} | Return 1 |
| rF(3) | {6,?,1},{5,?,4},{3,2,4} | Return 1 + 1 = 2 |
| rF(5) | {6,?,1},{5,5,4} | Return 2 + 3 = 5 |
| rF(6) | {6,?,1} | Call rF(4) |
| rF(4) | {6,?,1},{4,?,4} | Push AR |
| rF(3) | {6,?,1},{4,?,4},{3,?,4} | Push AR |
| rF(2) | {6,?,1},{4,?,4},{3,?,4},{2,?,4} | Push AR |
| rF(2) | {6,?,1},{4,?,4},{3,?,4},{2,1,4} | Return 1 |
| rF(3) | {6,?,1},{4,?,4},{3,?,4} | Call rF(1) |
| rF(1) | {6,?,1},{4,?,4},{3,?,4},{1,?,4} | Push AR |
| rF(1) | {6,?,1},{4,?,4},{3,?,4},{1,1,4} | Return 1 |
| rF(3) | {6,?,1},{4,?,4},{3,2,4} | Return 1 + 1 = 2 |
| rF(4) | {6,?,1},{4,?,4} | Call rF(2) |
| rF(2) | {6,?,1},{4,?,4},{2,?,4} | Return 1 |
| rF(2) | {6,?,1},{4,?,4},{2,1,4} | Return 1 |
| rF(4) | {6,?,1},{4,3,4} | Return 1 + 2 = 3 |
| rF(6) | {6,8,1} | Return 3 + 5 = 8 |
| Main Driver | {Whatever main stack had} | Assign 8 to x |

Project 8.1 #1



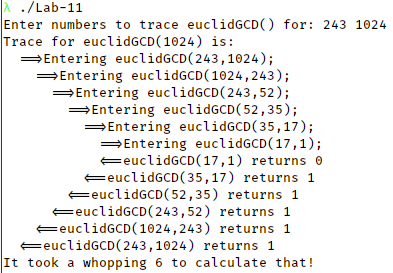
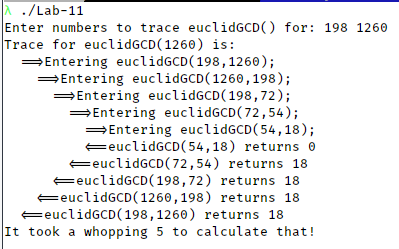
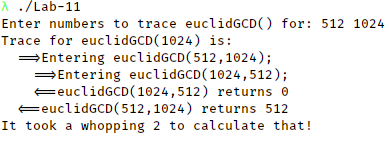
Project 8.1 #2:



**You asked for it:**

Enter a number to trace recFibonacci() for: 10   
Trace for recFibonacci(10) is:    
 ==>Entering recFibonacci(10);   
   ==>Entering recFibonacci(9);   
     ==>Entering recFibonacci(8);   
       ==>Entering recFibonacci(7);   
         ==>Entering recFibonacci(6);   
           ==>Entering recFibonacci(5);   
             ==>Entering recFibonacci(4);   
               ==>Entering recFibonacci(3);   
                 ==>Entering recFibonacci(2);   
                 <==recFibonacci(2) returns 1   
                 ==>Entering recFibonacci(1);   
                 <==recFibonacci(1) returns 1   
               <==recFibonacci(3) returns 2   
               ==>Entering recFibonacci(2);   
               <==recFibonacci(2) returns 1   
             <==recFibonacci(4) returns 3   
             ==>Entering recFibonacci(3);   
               ==>Entering recFibonacci(2);   
               <==recFibonacci(2) returns 1   
               ==>Entering recFibonacci(1);   
               <==recFibonacci(1) returns 1   
             <==recFibonacci(3) returns 2   
           <==recFibonacci(5) returns 5   
           ==>Entering recFibonacci(4);   
             ==>Entering recFibonacci(3);   
               ==>Entering recFibonacci(2);   
               <==recFibonacci(2) returns 1   
               ==>Entering recFibonacci(1);   
               <==recFibonacci(1) returns 1   
             <==recFibonacci(3) returns 2   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
           <==recFibonacci(4) returns 3   
         <==recFibonacci(6) returns 8   
         ==>Entering recFibonacci(5);   
           ==>Entering recFibonacci(4);   
             ==>Entering recFibonacci(3);   
               ==>Entering recFibonacci(2);   
               <==recFibonacci(2) returns 1   
               ==>Entering recFibonacci(1);   
               <==recFibonacci(1) returns 1   
             <==recFibonacci(3) returns 2   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
           <==recFibonacci(4) returns 3   
           ==>Entering recFibonacci(3);   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
             ==>Entering recFibonacci(1);   
             <==recFibonacci(1) returns 1   
           <==recFibonacci(3) returns 2   
         <==recFibonacci(5) returns 5   
       <==recFibonacci(7) returns 13   
       ==>Entering recFibonacci(6);   
         ==>Entering recFibonacci(5);   
           ==>Entering recFibonacci(4);   
             ==>Entering recFibonacci(3);   
               ==>Entering recFibonacci(2);   
               <==recFibonacci(2) returns 1   
               ==>Entering recFibonacci(1);   
               <==recFibonacci(1) returns 1   
             <==recFibonacci(3) returns 2   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
           <==recFibonacci(4) returns 3   
           ==>Entering recFibonacci(3);   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
             ==>Entering recFibonacci(1);   
             <==recFibonacci(1) returns 1   
           <==recFibonacci(3) returns 2   
         <==recFibonacci(5) returns 5   
         ==>Entering recFibonacci(4);   
           ==>Entering recFibonacci(3);   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
             ==>Entering recFibonacci(1);   
             <==recFibonacci(1) returns 1   
           <==recFibonacci(3) returns 2   
           ==>Entering recFibonacci(2);   
           <==recFibonacci(2) returns 1   
         <==recFibonacci(4) returns 3   
       <==recFibonacci(6) returns 8   
     <==recFibonacci(8) returns 21   
     ==>Entering recFibonacci(7);   
       ==>Entering recFibonacci(6);   
         ==>Entering recFibonacci(5);   
           ==>Entering recFibonacci(4);   
             ==>Entering recFibonacci(3);   
               ==>Entering recFibonacci(2);   
               <==recFibonacci(2) returns 1   
               ==>Entering recFibonacci(1);   
               <==recFibonacci(1) returns 1   
             <==recFibonacci(3) returns 2   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
           <==recFibonacci(4) returns 3   
           ==>Entering recFibonacci(3);   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
             ==>Entering recFibonacci(1);   
             <==recFibonacci(1) returns 1   
           <==recFibonacci(3) returns 2   
         <==recFibonacci(5) returns 5   
         ==>Entering recFibonacci(4);   
           ==>Entering recFibonacci(3);   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
             ==>Entering recFibonacci(1);   
             <==recFibonacci(1) returns 1   
           <==recFibonacci(3) returns 2   
           ==>Entering recFibonacci(2);   
           <==recFibonacci(2) returns 1   
         <==recFibonacci(4) returns 3   
       <==recFibonacci(6) returns 8   
       ==>Entering recFibonacci(5);   
         ==>Entering recFibonacci(4);   
           ==>Entering recFibonacci(3);   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
             ==>Entering recFibonacci(1);   
             <==recFibonacci(1) returns 1   
           <==recFibonacci(3) returns 2   
           ==>Entering recFibonacci(2);   
           <==recFibonacci(2) returns 1   
         <==recFibonacci(4) returns 3   
         ==>Entering recFibonacci(3);   
           ==>Entering recFibonacci(2);   
           <==recFibonacci(2) returns 1   
           ==>Entering recFibonacci(1);   
           <==recFibonacci(1) returns 1   
         <==recFibonacci(3) returns 2   
       <==recFibonacci(5) returns 5   
     <==recFibonacci(7) returns 13   
   <==recFibonacci(9) returns 34   
   ==>Entering recFibonacci(8);   
     ==>Entering recFibonacci(7);   
       ==>Entering recFibonacci(6);   
         ==>Entering recFibonacci(5);   
           ==>Entering recFibonacci(4);   
             ==>Entering recFibonacci(3);   
               ==>Entering recFibonacci(2);   
               <==recFibonacci(2) returns 1   
               ==>Entering recFibonacci(1);   
               <==recFibonacci(1) returns 1   
             <==recFibonacci(3) returns 2   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
           <==recFibonacci(4) returns 3   
           ==>Entering recFibonacci(3);   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
             ==>Entering recFibonacci(1);   
             <==recFibonacci(1) returns 1   
           <==recFibonacci(3) returns 2   
         <==recFibonacci(5) returns 5   
         ==>Entering recFibonacci(4);   
           ==>Entering recFibonacci(3);   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
             ==>Entering recFibonacci(1);   
             <==recFibonacci(1) returns 1   
           <==recFibonacci(3) returns 2   
           ==>Entering recFibonacci(2);   
           <==recFibonacci(2) returns 1   
         <==recFibonacci(4) returns 3   
       <==recFibonacci(6) returns 8   
       ==>Entering recFibonacci(5);   
         ==>Entering recFibonacci(4);   
           ==>Entering recFibonacci(3);   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
             ==>Entering recFibonacci(1);   
             <==recFibonacci(1) returns 1   
           <==recFibonacci(3) returns 2   
           ==>Entering recFibonacci(2);   
           <==recFibonacci(2) returns 1   
         <==recFibonacci(4) returns 3   
         ==>Entering recFibonacci(3);   
           ==>Entering recFibonacci(2);   
           <==recFibonacci(2) returns 1   
           ==>Entering recFibonacci(1);   
           <==recFibonacci(1) returns 1   
         <==recFibonacci(3) returns 2   
       <==recFibonacci(5) returns 5   
     <==recFibonacci(7) returns 13   
     ==>Entering recFibonacci(6);   
       ==>Entering recFibonacci(5);   
         ==>Entering recFibonacci(4);   
           ==>Entering recFibonacci(3);   
             ==>Entering recFibonacci(2);   
             <==recFibonacci(2) returns 1   
             ==>Entering recFibonacci(1);   
             <==recFibonacci(1) returns 1   
           <==recFibonacci(3) returns 2   
           ==>Entering recFibonacci(2);   
           <==recFibonacci(2) returns 1   
         <==recFibonacci(4) returns 3   
         ==>Entering recFibonacci(3);   
           ==>Entering recFibonacci(2);   
           <==recFibonacci(2) returns 1   
           ==>Entering recFibonacci(1);   
           <==recFibonacci(1) returns 1   
         <==recFibonacci(3) returns 2   
       <==recFibonacci(5) returns 5   
       ==>Entering recFibonacci(4);   
         ==>Entering recFibonacci(3);   
           ==>Entering recFibonacci(2);   
           <==recFibonacci(2) returns 1   
           ==>Entering recFibonacci(1);   
           <==recFibonacci(1) returns 1   
         <==recFibonacci(3) returns 2   
         ==>Entering recFibonacci(2);   
         <==recFibonacci(2) returns 1   
       <==recFibonacci(4) returns 3   
     <==recFibonacci(6) returns 8   
   <==recFibonacci(8) returns 21   
 <==recFibonacci(10) returns 55

**Project 8.1 #4**

****

**Code: (Applies for all of them except for minor differences in main())**

/// \file Project8-1.cpp

/// \author Johnathan Lee for CSCI 1107

/// \date Due 04/10/18

#include <iostream>

using namespace std;

unsigned indent = 0; ///>! For tracing functions

unsigned numCalls = 0; ///>! For counting calls.

void **doIndent**(bool arrowRight);

int **f**(int n) {

int retVal;

indent += 2;

doIndent(true);

cout << "Entering f(" << n << ");\n";

if (n < 2)

retVal = 0;

else

retVal = 1 + f(n / 2);

doIndent(false);

cout << "f(" << n << ") returns " << retVal << "\n";

indent -= 2;

return retVal;

}

int **recFibonacci**(int n);

int **euclidGCD**(int a, int b);

int **main**() {

int m, n;

string funcName = "euclidGCD";

cout << "Enter numbers to trace " << funcName << "() for: ";

cin >> m >> n;

cout << "Trace for " << funcName << "(" << n << ") is: \n";

euclidGCD(m, n);

cout << "It took a whopping " << numCalls << " to calculate that!\n";

}

/// \brief Helper function for tracing

/// \param arrowRight Should we use a ==> ? If not, use a <==.

void **doIndent**(bool arrowRight) {

for (int i = 0; i < indent; i++)

cout << " ";

cout << (arrowRight ? "==>" : "<==");

}

/// \brief Recursively calculates the fibonacci number for n

/// \param n The index into the fibonacci sequence to calculate.

/// \returns The value of the fibonacci sequence at index n.

/// \note Don't do this for n > 30 or so... Just don't.

int **recFibonacci**(int n) {

{ // Tracing stuff

numCalls++;

indent += 2;

doIndent(true);

cout << "Entering recFibonacci(" << n << ");\n";

}

int ret;

if (n == 1 || n == 2) // 2

ret = 1; // 3

else

ret = recFibonacci(n - 1) + recFibonacci(n - 2); // 4

{ // Tracing stuff

doIndent(false);

cout << "recFibonacci(" << n << ") returns " << ret << "\n";

indent -= 2;

}

return ret;

}

/// \brief Calculates the greatest common divisor of a and b using the Euclidean

/// Algorithm.

/// \param a see next

/// \param b Together with a, the numbers whose GCD we will find.

/// \returns The GCD of a and b.

int **euclidGCD**(int a, int b) {

{ // Tracing stuff

numCalls++;

indent += 2;

doIndent(true);

cout << "Entering euclidGCD(" << a << "," << b << ");\n";

}

int ret;

int r = a % b;

if (r == 0) {

ret = 0;

} else {

int next = euclidGCD(b, r);

ret = (next == 0 ? r : next);

}

{ // Tracing stuff.

doIndent(false);

cout << "euclidGCD(" << a << "," << b << ") returns " << ret << "\n";

indent -= 2;

}

return ret;

}