/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brett Huffman

\* CMPSCI 4250

\* Jul 23, 2021

\* Project 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Program Code

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <stdio.h>

#include <stdlib.h>

#define DEFAULT\_SIZE 1000

void f1()

{

char lArray[DEFAULT\_SIZE] = {0};

static int n = 0;

static long int addr = (long) &lArray;

n++;

if(n > 10)

return;

else

{

// Print the Call # and address of array

printf("\nCall: #%d @ %p", n, lArray);

// Next get the current address of array

// As a long. This prevents a warning

long int addrCurrent = (long) &lArray;

// Debugging

// printf("\nCurrent: #%ld", addrCurrent);

// printf("\nPrior: : #%ld", addr);

// Print the difference

printf("\nAR Size: #%d - %d\n", n, (addr-addrCurrent));

// Store for next time

addr = (long) &lArray;

// Call recursively

f1();

}

}

void f2()

{

// Get address of the very start of the stack

void \*pStart;

static long int addrStart = (long)(&pStart);

char lArray[DEFAULT\_SIZE] = {0};

static int n = 0;

static long int addr = (long) &lArray;

n++;

// There is no terminating clause in this

// one - it's going to seg fault

// Print the Call # and address of array

printf("\nCall: #%d @ %p", n, lArray);

// Next get the current address of array

// As a long. This prevents a warning

long int addrCurrent = (long) &lArray;

// Debugging

// if(n > 10)

// return;

// printf("\nCurrent: #%ld", addrCurrent);

// printf("\nPrior: : #%ld", addr);

// Print the difference

printf("\nAR Size: #%d - %d\n", n, (addr-addrCurrent));

// Print the ongoing total size of the stack

printf("Stack Size: #%d - %d\n", n, (addrStart-addrCurrent));

// Store for next time

addr = (long) &lArray;

// Call recursively

f2();

}

void f3()

{

char \*lArray = (char\*)malloc( sizeof(char) \* DEFAULT\_SIZE );

static int n = 0;

static long int addr = (long) &lArray;

n++;

if(n > 10)

return;

else

{

// Print the Call # and address of array

printf("\nCall: #%d @ %p", n, lArray);

// Next get the current address of array

// As a long. This prevents a warning

long int addrCurrent = (long) &lArray;

// Debugging

// printf("\nCurrent: #%ld", addrCurrent);

// printf("\nPrior: : #%ld", addr);

// Print the difference

printf("\nAR Size: #%d - %ld\n", n, (addr-addrCurrent));

// Store for next time

addr = (long) &lArray;

// Call recursively

f3();

// Free the memory

free(lArray);

}

}

int main() {

f1();

//f2();

//f3();

return 0;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function 1 Run

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Call: #1 @ 0x7ffee5089c50

AR Size: #1 - 0

Call: #2 @ 0x7ffee5089840

AR Size: #2 - 1040

Call: #3 @ 0x7ffee5089430

AR Size: #3 - 1040

Call: #4 @ 0x7ffee5089020

AR Size: #4 - 1040

Call: #5 @ 0x7ffee5088c10

AR Size: #5 - 1040

Call: #6 @ 0x7ffee5088800

AR Size: #6 - 1040

Call: #7 @ 0x7ffee50883f0

AR Size: #7 - 1040

Call: #8 @ 0x7ffee5087fe0

AR Size: #8 - 1040

Call: #9 @ 0x7ffee5087bd0

AR Size: #9 - 1040

Call: #10 @ 0x7ffee50877c0

AR Size: #10 - 1040

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function 2 Run

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Call: #7929 @ 0x7ffeeef4cd50

AR Size: #7929 - 1056

Stack Size: #7929 - 8371960

Call: #7930 @ 0x7ffeeef4c930

AR Size: #7930 - 1056

Stack Size: #7930 – 8373016

Call: #7931 @ 0x7ffeeef4c510

AR Size: #7931 - 1056

Stack Size: #7931 - 8374072

Call: #7932 @ 0x7ffeeef4c0f0

AR Size: #7932 - 1056

Stack Size: #7932 - 8375128

Call: #7933 @ 0x7ffeeef4bcd0

AR Size: #7933 - 1056

Stack Size: #7933 - 8376184

Call: #7934 @ 0x7ffeeef4b8b0

AR Size: #7934 - 1056

Stack Size: #7934 - 8377240

Call: #7935 @ 0x7ffeeef4b490

AR Size: #7935 - 1056

Stack Size: #7935 - 8378296

Call: #7936 @ 0x7ffeeef4b070

AR Size: #7936 - 1056

Stack Size: #7936 - 8379352

Call: #7937 @ 0x7ffeeef4ac50

AR Size: #7937 - 1056

Stack Size: #7937 - 8380408

Call: #7938 @ 0x7ffeeef4a830

AR Size: #7938 - 1056

Stack Size: #7938 - 8381464

[1] 58366 segmentation fault ./main

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function 3 Run

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Call: #1 @ 0x7f85b1c057c0

AR Size: #1 - 0

Call: #2 @ 0x7f85b1c05bb0

AR Size: #2 - 48

Call: #3 @ 0x7f85b1c05fa0

AR Size: #3 - 48

Call: #4 @ 0x7f85b1c06390

AR Size: #4 - 48

Call: #5 @ 0x7f85b1c06780

AR Size: #5 - 48

Call: #6 @ 0x7f85b1c06b70

AR Size: #6 - 48

Call: #7 @ 0x7f85b1c06f60

AR Size: #7 - 48

Call: #8 @ 0x7f85b1c07350

AR Size: #8 - 48

Call: #9 @ 0x7f85b1c07740

AR Size: #9 - 48

Call: #10 @ 0x7f85b1c07b30

AR Size: #10 - 48