```
# File:     recursive.c
# Purpose:  Print values of the function R(i,j) defined by
#           the formulas
#
#             R(i,j) = i - j, if i or j is < 0
#             R(i,j) = R(i-1, j) + R(i, j-1), otherwise
#
# Compile:  gcc -g -Wall -o recursive recursive.c
# Run:      ./recursive <n>
#
# Cprototype:         long Recur (long i, long j)
# Args:               i = rdi, j = rsi
# Return val:Recursive(i, j) = rax
#

        .section .text
        .global Recursive

Recursive:
                        push    %rbp
                        mov     %rsp,  %rbp         # We need to store i, j and a return
                        sub     $24,   %rsp         # val from recursive call

                        # test if i < 0
                        cmp     $0,     %rdi        # If rdi = i < 0?
                        jl      sub_i_j             # Look at the flags register to see
whether
                                                    # the previous comparison result is < 0

                        # test if j < 0
                        cmp     $0,     %rsi        # If rsi = j < 0?
                        jl      sub_i_j             # Look at the flags register to see
whether
                                                    # the previous comparison result is < 0


                        mov     %rdi,   16(%rsp)    # Save i = rdi on the stack
                        mov     %rsi,   8(%rsp)     # Save j = rsi on the stack
                        sub     $1,     %rdi        # i = i - 1
                        call    Recursive
                        mov     %rax,  0(%rsp)      # Save t1 = Recursive(i-1, j) on the
stack
                        mov     16(%rsp),  %rdi     # Retrieve i
                        sub     $1,     %rsi        # j = j - 1
                        call    Recursive
                        add     0(%rsp),  %rax      # Return Recursive(i-1, j) + Recursive(i,
j-1)
```

```
sub_i_j:
                mov     %rdi,   %rax        # Save rdi = i to return val
                sub     %rsi,   %rax        # Return rax = rax - rsi = i - j
                jmp             done

done:
                leave                       # Assigns rbp to rsp
                ret
```