The following is an example of a MAL program containing two functions, namely `main` and `sum`. The function `sum` has one parameter and also a return value. Both C and MAL versions of these functions are given below.

### C Version

```c
void main(void) { int sum (int x) {
    int r = 10;
    int y = sum(r) + 9;
    printf("%d\n", y);
} return 0;
```

### MAL Version

```mal
#Function: main
#The variables r and y are stored in $5 and $6 respectively.
#The function sum returns the value in $7.

.data
newline: .asciiz "\n"

.text
.globl main
main:
li $5, 10 #r is assigned 10.
jal sum
addi $6, $7, 9 #Compute y in $6.
move $a0, $6
li $v0, 1 #To print the value of y.
syscall
la $a0, newline #To print '\n'.
li $v0, 4
syscall
li $v0, 10 #To stop program.
syscall

(over)
```
#Function: sum
#The value of the parameter (x) is in $5. The return value must be in $7.

.text
sum:    bgtz $5, compute  #Compute return value if $5 contains
       #a value > 0.
   li    $7, 0
   jr    $31  #Else, return the value of 0.

#Need to compute the return value.
#Save and restore $6 and $8 so that they can be used as holders of
#temporary values during the computation.
compute:  sw    $6, 0($sp)  #Push $6 on stack.
   sw    $8, -4($sp)  #Push $8 on stack.
   addi  $sp, $sp, -8
   addi  $6, $5, 1   #$6 has the value x+1.
   mul   $6, $6, $5  #$6 has the value x(x+1).
   li    $8, 2
   div   $7, $6, $8  #$7 has the value x(x+1)/2
       #(i.e., the correct return value).
   lw    $8, 4($sp)  #Restore $8.
   lw    $6, 8($sp)  #Restore $6.
   addi  $sp, $sp, 8  #Pop stack (twice).
   jr    $31  #Return to main.