CS240
Laboratory 9 Assignment
X86 Stack

Inspect the following C program:

```
#include <stdio.h>

  long getAndSumValues() {
    long x;
    printf("Enter an integer: ");
    scanf(" %ld",&x);
    if (x == 0) {
      return 0;
    } else {
      return x + getAndSumValues();
    }
 }

int main() {
   long result;
   result = getAndSumValues();
   printf("The result = %ld\n",result);
   return 0;
}
```

It produces the following x86 instructions when compiled:

```
Dump of assembler code for function main:
//setup
  0x000000000040058a <+0>:      push   %rbp
  0x000000000040058b <+1>:      mov    %rsp,%rbp
  0x000000000040058e <+4>:      sub    $0x10,%rsp
  0x0000000000400592 <+8>:      mov    $0x0,%eax

//call getAndSumValues
  0x0000000000400597 <+13>:     callq  0x400534 <getAndSumValues>
  0x000000000040059c <+18>:     mov    %rax,-0x8(%rbp)  //store result

//print the result
  0x00000000004005a0 <+22>:     mov    $0x4006cd,%eax
  0x00000000004005a5 <+27>:     mov    -0x8(%rbp),%rdx
  0x00000000004005a9 <+31>:     mov    %rdx,%rsi
  0x00000000004005ac <+34>:     mov    %rax,%rdi
  0x00000000004005af <+37>:     mov    $0x0,%eax
  0x00000000004005b4 <+42>:     callq  0x400418 <printf@plt>
```

```
//set return value to 0 and finish
   0x00000000004005b9 <+47>:    mov    $0x0,%eax
   0x00000000004005be <+52>:    leaveq
   0x00000000004005bf <+53>:    retq
End of assembler dump.


Dump of assembler code for function getAndSumValues
//setup
   0x0000000000400534 <+0>:     push   %rbp
   0x0000000000400535 <+1>:     mov    %rsp,%rbp
   0x0000000000400538 <+4>:     sub    $0x10,%rsp

// prompt user to enter a value
   0x000000000040053c <+8>:     mov    $0x4006b8,%eax
   0x0000000000400541 <+13>:    mov    %rax,%rdi
   0x0000000000400544 <+16>:    mov    $0x0,%eax
   0x0000000000400549 <+21>:    callq  0x400418 <printf@plt>

 //accept the input from the user
   0x000000000040054e <+26>:    mov    $0x4006c8,%eax
   0x0000000000400553 <+31>:    lea    -0x8(%rbp),%rdx
   0x0000000000400557 <+35>:    mov    %rdx,%rsi
   0x000000000040055a <+38>:    mov    %rax,%rdi
   0x000000000040055d <+41>:    mov    $0x0,%eax
   0x0000000000400562 <+46>:    callq  0x400438 <scanf@plt

//explain this section
   0x0000000000400567 <+51>:    mov    -0x8(%rbp),%rax
   0x000000000040056b <+55>:    test   %rax,%rax
   0x000000000040056e <+58>:    jne    0x400577 <getAndSumValues+67>
   0x0000000000400570 <+60>:    mov    $0x0,%eax
   0x0000000000400575 <+65>:    jmp    0x400588 <getAndSumValues+84>
   0x0000000000400577 <+67>:    mov    $0x0,%eax
   0x000000000040057c <+72>:    callq  0x400534 <getAndSumValues>
   0x0000000000400581 <+77>:    mov    -0x8(%rbp),%rdx
   0x0000000000400585 <+81>:    add    %rdx,%rax


//finish
   0x0000000000400588 <+84>:    leaveq
   0x0000000000400589 <+85>:    retq
End of assembler dump.
```

1. What type of program is this (use the descriptive term you should be familiar with from other CS courses)?

2. Explain the highlighted section of code in some detail.

3. How is the stack used to accumulate the result?

4. What register is used to represent the variable *result?*

5. How is the *result* returned from *getAndSumValues* to the main program?