COMP.2030 LAB 11

NAME _____

1. A recursive C function, `recur`, is declared as

```
int recur(long *x, long y)
```

is compiled into the x86 code on the right.
Complete the C code of the function `recur` below.

```
int recur(long *x, long y) {
```

```
recur: pushq     %rbp
       movq      %rsp, %rbp
       subq      $16, %rsp
       movq      %rdi, -8(%rbp)
       movq      %rsi, -16(%rbp)
       cmpq      $0, -8(%rbp)
       jne       .L2
       movl      $-1, %eax
       jmp       .L3
.L2:
       cmpq      $10, -16(%rbp)
       jne       .L4
       movl      $0, %eax
       jmp       .L3
.L4:
       movq      -8(%rbp), %rax
       movq      (%rax), %rax
       cmpq      -16(%rbp), %rax
       jle       .L5
       movq      -8(%rbp), %rax
       addq      $8, %rax
       movq      (%rax), %rax
       movq      -16(%rbp), %rdx
       movq      %rdx, %rsi
       movq      %rax, %rdi
       call      recur
       addl      %eax, %eax
       jmp       .L3
.L5:
       movl      $1, %eax
.L3:
       leave
       ret
```

2. The function `long switch_prob(long x, long n)` is disassembled as shown below.

```
    long switch_prob(long x, long n)
    x in %rdi, n in %rsi
1   0000000000400590 <switch_prob>:
2     400590:  48 83 ee 3c              sub    $0x3c,%rsi
3     400594:  48 83 fe 05              cmp    $0x5,%rsi
4     400598:  77 29                    ja     4005c3 <switch_prob+0x33>
5     40059a:  ff 24 f5 f8 06 40 00     jmpq   *0x4006f8(,%rsi,8)
6     4005a1:  48 8d 04 fd 00 00 00     lea    0x0(,%rdi,8),%rax
7     4005a8:  00
8     4005a9:  c3                       retq
9     4005aa:  48 89 f8                 mov    %rdi,%rax
10    4005ad:  48 c1 f8 03              sar    $0x3,%rax
11    4005b1:  c3                       retq
12    4005b2:  48 89 f8                 mov    %rdi,%rax
13    4005b5:  48 c1 e0 04              shl    $0x4,%rax
14    4005b9:  48 29 f8                 sub    %rdi,%rax
15    4005bc:  48 89 c7                 mov    %rax,%rdi
16    4005bf:  48 0f af ff              imul   %rdi,%rdi
17    4005c3:  48 8d 47 4b              lea    0x4b(%rdi),%rax
18    4005c7:  c3                       retq
```

The jump table resides in a different area of memory. We can see from the indirect jump on line 5 that the jump table begins at address `0x4006f8`. Using the GDB debugger, we can examine the six 8-byte words of memory comprising the jump table with the command `x/6gx 0x4006f8`. GDB prints the following:

```
(gdb) x/6gx 0x4006f8
0x4006f8:    0x00000000004005a1    0x00000000004005c3
0x400708:    0x00000000004005a1    0x00000000004005aa
0x400718:    0x00000000004005b2    0x00000000004005bf
```

Fill in the body of the `switch` statement with C code that will have the same behavior as the machine code.

```
long switch_prob(long x, long n) {
      long result = x;
      switch(n) {
```

3. The following code transposes the elements of an MxM array, where M is a constant defined by `#define`. When compiled, gcc generates the assembly code for the inner loop of the function as shown on the right.

```
void transpose(long A[M][M]) {
    long i, j;
    for (i = 0; i < M; i++)
        for (j = 0; j < i; j++) {
            long t = A[i][j];
            A[i][j] = A[j][i];
            A[j][i] = t;
        }
}
```

```
.L6:
  movq    (%rdx), %rcx
  movq    (%rax), %rsi
  movq    %rsi, (%rdx)
  movq    %rcx, (%rax)
  addq    $8, %rdx
  addq    $120, %rax
  cmpq    %rdi, %rax
  jne     .L6
```

A. Which register holds a pointer to array element `A[i][j]`?

B. Which register holds a pointer to array element `A[j][i]`?

C. What is the value of M?