Answer Sheet

Name: Student ID: TA: ______

1. Here we go.

1. (1) 0x0(,%rax,4) / (,%rax,4) (4) $0x7/$7

   (2) lea/leaq (5) da/DA

   (3) -0x34(%rbp)/-52(%rbp)

2. 0x200000000

2. Let's jump!

1. (1) case 0x23: **or** case 35: (5) ret = ret + 2;

   (2) default: (6) case 0x21: **or** case 33:

   (3) ret = ret << 2; (7) case 0x20: **or** case 32:

   (4) case 0x24:case 0x25: **or** case 36:case 37:

2. Line 12 **or** 13

3.

① Line Number: 10. Reason: If the source operand is a memory operand, then regardless of the condition, the memory operand is read. For the condition `yp == NULL`, `cmovne` used here would access invalid address 0x0 and generates segmentation fault. To avoid this situation conditional jump should be used instead of conditional move.

② Line Number: 29. Reason: The instruction `movzlq` does not exist (please refer the paragraph under **graph 3.6** in CSAPP) and `movl` itself will fill the destination register's upper 4 bytes with zeros. The correct assembly should be `movl %eax, %edx`.

3. It is the time to hack!

1. (1) **`key[i]`** (4) **`src[i] + key[i] * i`**

   (2) **`key[i]* i`** (5) **`dst[i-1]`**

(3) **src[i]**

2. `if ( i % 2 == 0) goto branch1;`

3.

	0	1	2	3	4
Src	0x73	0x01	0x02	0x03	0x04

4. What the hack is it?

1. (1) l <= h && arr[l] <= arr[p] (2) l <= h && arr[h] > arr[p]

   (3)
   ```
   int tmp = arr[l];
   arr[l] = arr[h];
   arr[h] = tmp;
   ```

   (4)
   ```
   int tmp = arr[p];
   arr[p] = arr[h];
   arr[h] = tmp;
   ```

2. partition