Xi Liu, xl3504, Homework 2

1. 1.

```
"prev" = "previous"
"ret" = "return"
"addr" = "address"

|    ... |
|  prev %rbp before main |
|  line after L7, or ret addr (current %rip, addr of current instruction line) |
|  prev %rbp before comp | <- frame pointer (%rbp)
|  0, or value of e |
|  0, or value of f |  <- stack pointer (%rsp)
```

1.2.

```
|    ... |
|  prev %rbp before main |
|  line after L7, or ret addr (current %rip, addr of current instruction line) |
|  prev %rbp before comp | <- frame pointer (%rbp)
|  0, or value of e |
|  0, or value of f |  <- stack pointer (%rsp)
```

%rip would correspond to L5

1.3.

```
|    ... |
|  prev %rbp before main |
|  line after L7, or ret addr (current %rip, addr of current instruction line) |
|  prev %rbp before comp |
|  0, or value of e |
|  0, or value of f |
```

```
|  L5, or ret address (current %rip, address of current line of instruction) |
|  prev %rbp before mul | <- frame pointer (%rbp) <- stack pointer (%rsp)
```

1.4

```
|    ... |
|  prev %rbp before main |
|  line after L7, or ret addr (current %rip, addr of current instruction line) |
|  prev %rbp before comp |
|  0, or value of e |
|  0, or value of f |
|  L5, or ret addr (current %rip, addr of current instruction line) | <- frame po:
```

%rip would correspond to L1

2.1

```
node_t *
find_insert_pos(node_t *head, node_t *node)
{
    if (head == NULL) return NULL;

    node_t *ret = NULL;

    // 2.1 your code here
    if(node -> id < head -> id)
    {
        return NULL;
    }

    ret = head;
    while(ret)
    {
        if( ret -> id >= node -> id )
```

2

```
            {
                return ret;
            }
            if(ret -> next -> id > node -> id)
            {
                return ret;
            }
            ret = ret -> next;
        }

        return ret;
    }
```

2.2

```
node_t *
 list_insert(node_t *head, node_t *node)
 {
     if (head == NULL) return node;

     // find the proper position to insert this node pair.
     node_t *pos = find_insert_pos(head, node);

     // 2.2 your code here

     if(!pos)
     {
         node -> next = head;
         head = node;
     }
     else
     {
```

```
            node -> next = pos -> next;
            pos -> next = node;
        }
    }
```

3.1

i.   echo hello
ii. echo hello $world
iii. echo hello
iv. hello
v. -bash: syntax error near unexpected token `echo'

3.2
i. hello world
ii. no printed message seen
iii. hello world

3.3
i.
a
b

ii.
a
b

iii
[1] 1941
b
a

3.4

on my computer, below seems to output correctly:

first part

```
cat members.txt | head -n100| cut -d ':' -f2
```

second part

```
cat members.txt | sort -f | head -n100| cut -d ':' -f2 | tee names.txt
```

grep might be needed:

first part

```
cat members.txt |  grep "^Name:[a-zA-Z']\+$" | head -n100| cut -d ':' -f2
```

second part

```
cat members.txt |  grep "^Name:[a-zA-Z']\+$" | sort -f | head -n100| cut -d ':' -f
```