计算机程序的构造与解释（SICP）
期中试题参考答案

1. **(18 points) What Would Python Display**

Expression	Interactive Output	
`2**11 - 28` `True and not 1/0`	2020 Error	(2 points)

`min, max = max, min(2, 0)`
`print(min)`
`print(max)`
	Function
	0
		(2 points)

`t = (0, [1])`
`t[0] = 2`
`s = t[1]`
`t[1].append(2)`
`s is t[1]`
`t[1][1:] + [t[0]] + [2, 0]`
	Error
	True
	[2, 0, 2, 0]
		(3 points)

`foo('SICP', print)(2020)`	SICP 2020	(2 points)

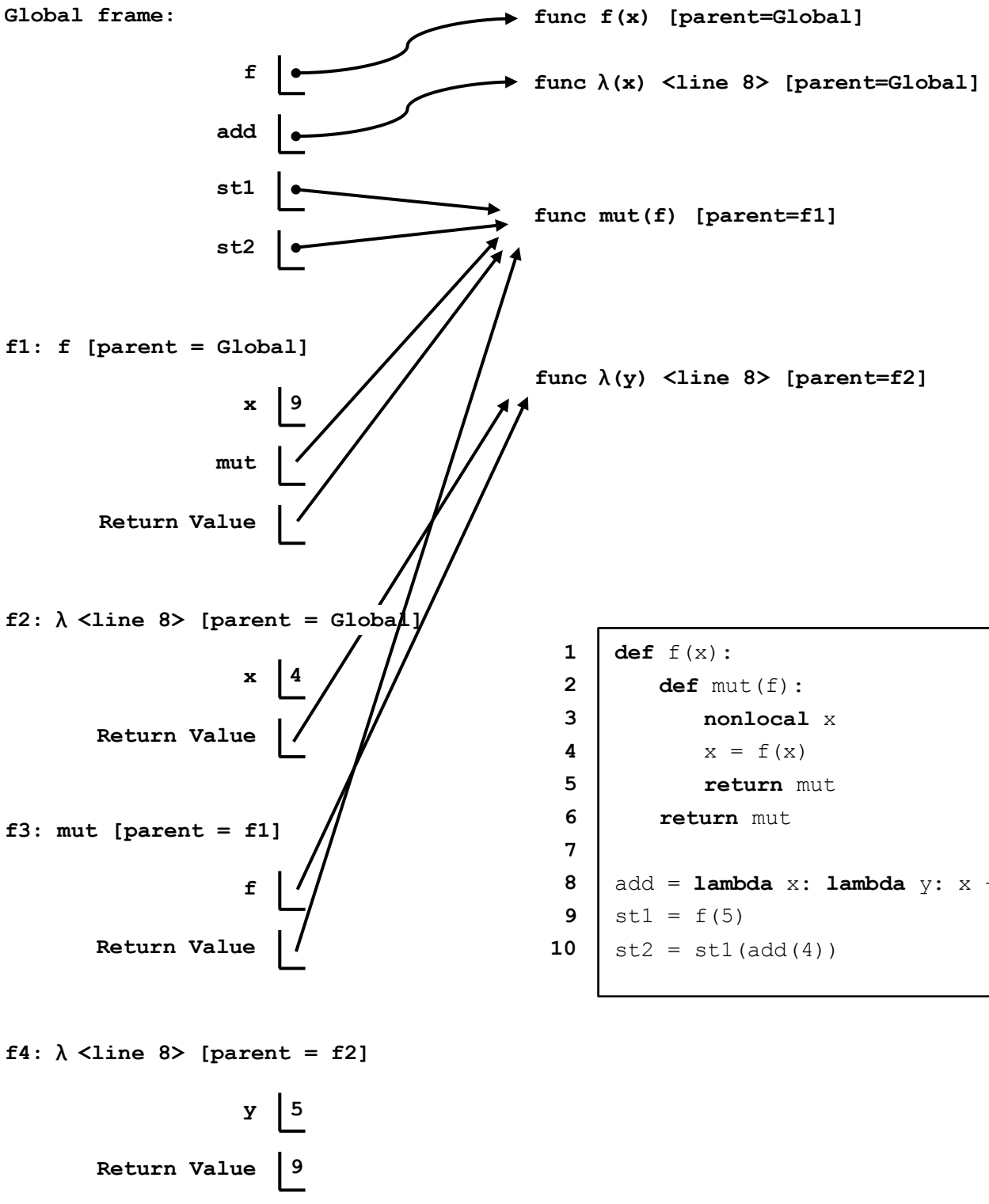
`my_all(True, False)`
`my_all([])`
`my_all([True, False])`
	Error
	True
	False
		(3 points)

`p = gen_p()`
`for _ in range(3):`
`  print(next(p))`
`p is gen_p()`
	2
	3
	5
	False
		(4 points)

`com(3)(m10)(a2)(m10)(20)`	2020	(2 points)

评分细则：每行输出计 1 分。本题大小写错误或是链表少逗号等笔误均不给分。如果有冗余的输出，将冗余输出视为-1 分。

2. **(12 points) Environment Diagram** (0.5 points each box or blank)




```
1  def f(x):
2      def mut(f):
3          nonlocal x
4          x = f(x)
5          return mut
6      return mut
7
8  add = lambda x: lambda y: x + y
9  st1 = f(5)
10 st2 = st1(add(4))
```

评分细则：所有横线填空（变量名、函数名、parent）都计 0.5 分，必须完全写正确。所有引用根据引用的对象是否正确给分，也都计 0.5 分，必须完全写正确才给分。f1 里面的 x 和 mut 如果位置反了，不扣分。

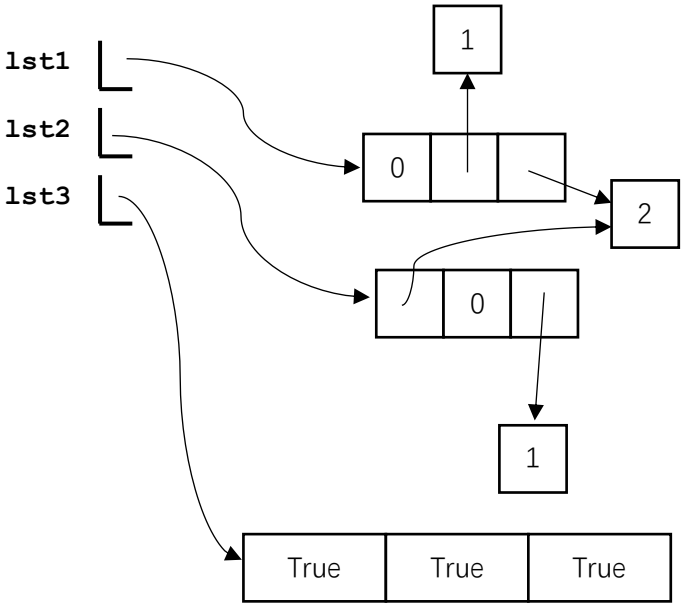
3. **(12 points) Boxing Day**

(a) (**3 points**)

```
lst1 = [0, [1], [2]]
lst2 = [lst1[2], 0, lst1[1][:]]
lst3 = [x in lst2 for x in lst1]
```

lst1、lst2——均画对才给 1 分

lst3——2 分

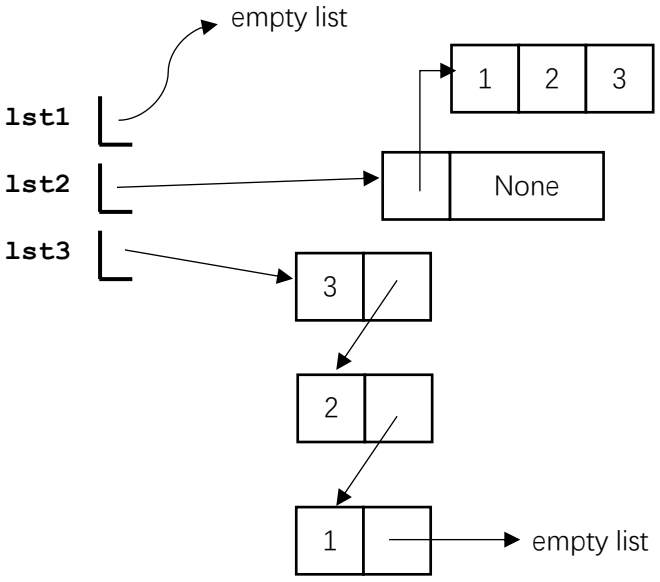


(b) **(4 points)**

```
lst1 = [1, 2]
lst2 = [lst1, lst1.append(3)]
lst3 = []
while lst1:
    lst3 = [lst1[0], lst3]
    lst1 = lst1[1:]
```

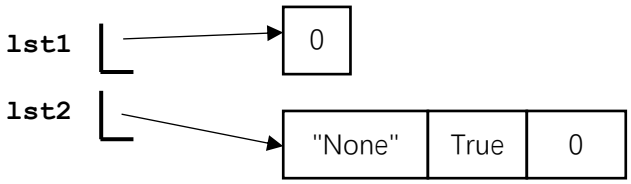
lst1、lst2 各 1 分

lst3——2 分



(c) **(2 points)**

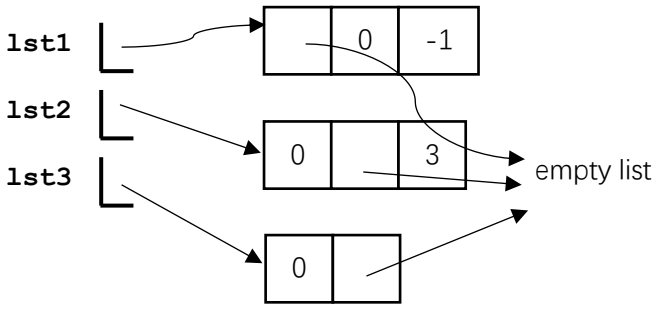
```
lst1 = [-1, 0, True, 'None']
lst2 = []
lst1, k = lst1[1:], lst1[0]
popper = lambda lst:
  lambda: lst[k] and lst.pop()
popper = popper(lst1)
for _ in range(len(lst1)):
    lst2.append(popper())
```



lst1、lst2 各 1 分

(d) **(3 points)**

```
lst1 = [[2], 0]
lst2 = lst1[1:] + [lst1[0]]
lst3 = list(lst2)
for item in lst3:
    if item:
        lst2.append(item.pop() + 1)
    else:
        item -= 1
        lst1.append(item)
```



lst1、lst2、lst3 各 1 分

4. (5 points) Reverse Digits

```
def reverse_digit(n):
    result = 0
    while n > 0:
        result = 10 * result + n % 10
        n = n // 10
    return result
```

评分细则：每空一分，等价的写法自行判断。

5. **(5 points) Simple Math (NOT Church numeral!)**

```
plus = lambda x, y: x + y # auxiliary function used to define mult
def mult(m, n):
    """
    mult(m, n) returns the value of m*n, where both m and n are positive integers.
    Just put one name or constant in each of the blank space.
    Do NOT write anything more.
    """
    if m == 0:
        return 0
    m1 = m - 1
    return plus(n, (mult(m1, n)))
```

评分细则：每空一分，按题目规定，只能写 name 或 constant，违反则不给分。

6. **(27 points) Trees**

(a, 4 points)

```
t1 = tree(5, [tree(2, [tree(3), tree(6)]), tree(1), tree(4)])
```

评分细则：若仅有括号不匹配等笔误，给 3 分，其他错误不给分。

(b, 6 points)

```
def label_sum(t):
    if is_leaf(t):
        return label(t)
    return label(t) + sum([label_sum(b) for b in branches(t)])
```

评分细则：每空一分，等价的写法自行判断。

(c, 5 points) Define the function `reverse_tree(t)`, which forms a new tree, where the branch order of each node is the **reverse** of the branch order of the corresponding node on `t`. For instance, `reverse_tree(t1)` for the `t1` defined above generates the tree `t3`.

```
def reverse_tree(t):
    if is_leaf(t): # 1 point
```

```
        return tree(label(t))  # 1 point
    reversed_branches = [reverse_tree(branches(t)[i]) for i in
range(len(branches(t)) - 1, -1, -1)] # 2 points, "reversed branches =
[reverse tree(b) for b in branches(t)][::-1]" is also correct.
    return tree(label(t), reversed_branches)   # 1 point
```

评分细则：见参考代码。等价的写法自行判断。

(d, 12 points)

```
def label_sum(t):
    return fold_tree(t, lambda v: v, lambda v, vs: v + sum(vs))

def height(t):
    return fold_tree(t, lambda v: 1, lambda v, vs: 1 + max(vs))

def preorder(t):
    return fold_tree(t, lambda v: [v],
        lambda v, vs: reduce(lambda x, y: x + y, [v], vs))
```

评分细则：每个函数的第一空（base_func）计 1 分，第二空（merge_func）计 3 分。等价的写法自行判断。

7. (21 points) Automatic Function Composition

(a, 4 points)

(1) `[[div2, div2, inc], [div2, inc, div2], [inc, div2, div2]]` (3 points)

(2) `[[div3]]` (1 point)

评分细则：每个 solution 要完全写对，solution 内部的函数顺序不能写错，solution 之间的顺序可以调换。冗余的 solution 倒扣分（一个冗余的 solution 扣一分）

(b, 12 points) (1 point each blank, unless specifically annotated)

```
def satisfy(io_pair, sol):
    input_x = io_pair[0]
    output_y = io_pair[1]
    eval_result = input_x     # io pair[0] is also ok
    for func in sol:
        eval_result = func(eval_result)
    return eval_result == output_y   # io pair[1] is also ok
```

```
def afc(io_pair, func_list, times_list):
    assert len(func_list) == len(times_list)
    solutions = []

    def sol_search(remain_times_list, cur_sol):
        if cur_sol and satisfy(io_pair, cur_sol):  (2 points)
            solutions.append(cur_sol)
        for i in range(len(func_list)):
            if remain_times_list[i] > 0:
                new_times_list = [remain_times_list[j] if j != i \
                    else remain_times_list[j] - 1 for j in range(len(func_list))]
                (2 points)
                sol_search(new_times_list, cur_sol + [func_list[i]])    (2 points)

    sol_search(times_list, [])
    return solutions
```

评分细则：除了标注的三处分值为两分的空格，其余分值均为每空一分。等价的写法自行判断。

(c, 5 points)

```
def extended_afc(io_pair_list, func_list, times_list):
    assert len(io_pair_list) > 0
    candidate_solutions = afc(io_pair_list[0], func_list, times_list) (2 points)
    return my_filter(lambda sol: my_all([satisfy(io_pair, sol) \
        for io_pair in io_pair_list]), candidate_solutions)     (3 points)
```

评分细则：每空得分按照参考代码的要求。等价的写法自行判断。

8. (Extra 3 points) Questions about SICP

(a) The full name of SICP is Structure and Interpretation of Computer Programs. (in English)

(b) The two instructors' names are 这个不用我说了吧, respectively. (in Chinese)

(c) Write the names of at least two teaching assistants: 这个也不用我说了吧(in Chinese)

评分细则：英文课程名大小写写错、单复数写错不扣分，单词拼错不给分。中文名字写错不给分。(c)问只写对一个名字也不给分。若本题得分与前面得分相加有超过 100 分的情况，取 100 分为最终分数。