1. (6%) 以下代码为一个 80 年代的 BASIC 语言程序，请据此回答问题。

```
10 LET A = 18
20 LET B = 12
30 IF B=0 GOTO 80
40 LET C = A % B
50 LET A = B
60 LET B = C
70 GOTO 30
80 PRINT A
```

a) 这个程序是否是结构化的程序(1%)，为什么？(1%)

b) 将这段程序改写成 MUA 的程序，实现相同的计算结果。(4%)

2. (10%) 有以下语法，它能否产生语句A = B + C * (A + D)？如果能，给出左推导的过程，注意需要写出推导的每一步。

```
<assign> → <id> = <expr>
<id> → A | B | C | D
<expr> → <expr> + <term> | <term>
<term> → <term> * <factor> | <factor>
<factor> → ( <expr> ) | <id>
```

3. (5%) 名词解释以下术语（每个 1%）：

a) 变量；

b) 常量；

c) 字面量；

d) 形参；

e) 实参。

4. (8%) C 语言有#include，Java 语言和 Python 语言有 import。

a) 为什么编程语言通常需要这样的手段，它解决了什么问题？(2%)

b) 这三种语言所实现的方式有何不同，请分别解释每种语言的实现方式。(每个语言 2%)

5. (17%) 计算 x^n 的算法可以表达为下面的分段函数：

$$x^n = \begin{cases} 1; n = 0 \\ x^{\frac{n}{2}} \times x^{\frac{n}{2}}; n是偶数 \\ x \times x^{n-1}; n是奇数 \end{cases}$$

a) 请编写 MUA 程序以递归实现以上递归计算算法。(5%)

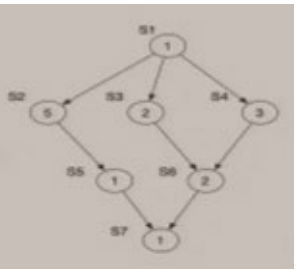
b) 这个递归计算是是线性递归还是树状递归？(1%)

c) 将 a)的递归实现的递归计算改成递归实现的迭代计算，给出 MUA 代码。(5%)

d) 如何识别出 c)所写的代码是一种迭代计算(2%)？

e) 如何优化 c)所写代码的迭代计算的解释执行？以你的 MUA 解释器的实现来说明。(4%)。

6. (8%) 根据如右计算图，回答：



a) 这个计算的 work 和 span 各是多少？(2%)

b) 核的数量不限时，理想并行加速比是多少？(2%)

c) 当系统只有两个核时，如何调度能有最高的并行加速性能（3%），此时的理想加速比是多少？(1%)

7. (8%) 函数调用时需要产生 Activation Record（AR），以存放函数所需的返回地址、参数和本地变量。C 语言的 AR 都是在堆栈中顺序分配的，新调用的函数的 AR 后进入堆栈。

a) 这样实现 AR 的优点是什么(2%)？

b) 这样实现 AR 的缺点是什么(2%)？

c) 你所实现的 MUA 的 AR 是怎样分配的？请给出较为详细的描述说明(4%)。

8. (10%) 以下 MUA 代码实现了[a,b]之间的累加：

```
make "sum [ [term a next b] [
    if gt :a :b
        [return 0]
        [return add term :a sum :term next :a :next :b]
]]
```

已知完美数的真约数之和正好等于自身。如 6=1+2+3，故 6 是一个完美数。以下代码可以判断数 x 是否为完美数，请将下划线部分填充完整：

```
if eq x sum _____
    [print "perfect]
    [print [not perfect]]
```

9. (9%) 以下 MUA 函数是小费马定理测试的核心代码：

```
make "fermat_test [ [n] [
    make "try_it [ [a] [
        return eq expmod :a :n :n :a]
    ]
    return try_it add rands… sub :n 1 1
]]
```

a) 其中的 try_it 函数符合函数式编程的哪个特性（闭包、柯里化、高阶函数还是内部函数）？(1%)

b) 简述上述特性的要点。(1%)

c) 根据今年的 MUA 语言定义，其中的 try_it 函数不能按照设计意图执行，为什么？(2%)

d) 如何改写这个 try_it 函数使得它能正确执行。(5%)

** 提示以下 MUA 操作：*

- *sentence <value1> <value2>：将 value1 和 value2 的元素合并成一个表*
- *list <value1> <value2>：将 value1 和 value2 合并为一个表*
- *join <list> <value>：将 value 作为 list 的最后一个元素加入到 list 中*

10. (19%) Java 的 Lamdba 表达式形如：(x)->{return x;}，已知 Java 的以下类和函数：

- IntStream.range(a,b)：返回一个表达[a，b）的流（Stream）；

- Stream.filter()：根据 Lambda 表达式的计算决定元素是否留在流中；
- Stream.map()：根据 Lambda 表达式的计算将流中的元素变换为新的值；
- Stream.sum()：将流中的数值累加起来返回单个值。

a) 利用以上类和函数，写一个 Java 函数 int sum_prime(int a, int b)，返回[a,b) 之间的所有的素数的和。在函数中只能写一句语句，也不能另外写函数或类。(5%)

b) 简述 a) 中代码的执行过程。(5%)

c) 如果已经实现了以下 MUA 函数：

- intstream :a :b，产生一个[a,b)的流；
- stream_filter :s :f，在流 s 上做过滤 f；
- stream_map :s :f，在流 s 上做映射 f；
- stream_sum :s，在流 s 上做累加。

不考虑流的数据表达形式，给出用 MUA 实现上述计算的代码。(5%)

d) 设计 c) 中的 MUA 的流的数据表达形式，用文字说明(4%)。