清华大学本科生考试试题 姓名_____ 班级_____ 学号_____

考试课程《软件分析与验证·样卷》 xxxx 年 x 月 x 日 （共 100 分）

（注：所有题目答案均写在答题本上，考试结束本试题纸也需要一并上交）

一、判断题

判断下列说法正确还是错误。

1. 若 P 是循环 **while** b **do** c 的循环不变式，则满足 $P \to b$。
   A. True
   B. False

   ➢ 参考答案：B

2. $\{x = y + 1\}$ **while** $x \neq 0 \wedge y \neq 0$ **do** $(x := x - 2;\ y := y - 2)$ $\{x = y + 1\}$.
   A. True
   B. False

   ➢ 参考答案：A

3. 不含有量词的T_{PA}是不可判定的。
   A. True
   B. False

   ➢ 参考答案：A

二、选择题

从以下各题所给的四个选项中选出**唯一**正确的一项。

4. 给定命题逻辑公式 F 和 G，下列说法**不正确**的是？
   A. 如果 F 和 G 都是不可满足的，则 $F \vee G$ 一定不可满足。
   B. 如果 F 是有效的且 G 不是有效的，则 $F \to G$ 一定不可满足。
   C. 如果 F 是可满足的且 $\neg G$ 是不可满足的，则 $F \wedge G$ 一定可满足。
   D. 如果 F 不是有效的且 G 是可满足的，则 $F \to G$ 一定可满足的。

   ➢ 参考答案：B

5. 下列关系**不是**良基关系的是？
   A. 非负有理数集合上的小于关系。
   B. 正整数集合上定义的序关系$\prec$，$a \prec b$ 当且仅当 a 整除 b 且 $a \neq b$。
   C. 任何一个有限结点的有向无环图上的关系R，$a\ R\ b$ 当且仅当存在有向边：$a \to b$。
   D. 自然数的有序对集合（$\mathbb{N} \times \mathbb{N}$）上定义的序关系$\prec$，$(n_1, n_2) \prec (m_1, m_2)$当且仅当$n_1 < m_1 \wedge n_2 < m_2$。

   ➢ 参考答案：A

6. $\mathrm{wlp}(\textbf{if } x \geq k_1 \textbf{ then } x := x - k_2 \textbf{ else } x := -x - k_2, x \geq k_3)$
   A. $(x \geq k_1 \to x - k_2 \geq k_3) \vee (x < k_1 \to -x - k_2 \geq k_3)$
   B. $(x \geq k_1 \to x + k_2 \geq k_3) \vee (x < k_1 \to -x - k_2 \geq k_3)$
   C. $(x \geq k_1 \to x - k_2 \geq k_3) \wedge (x < k_1 \to -x - k_2 \geq k_3)$
   D. $(x \geq k_1 \to x + k_2 \geq k_3) \wedge (x < k_1 \to -x - k_2 \geq k_3)$

➢ 参考答案：C

三、简答题

请根据要求完成以下题目，并提供必要的过程。

7. 给定如下的一阶逻辑公式 F，请回答其有效性，并利用相继式演算的方法进行证明。（提示：分别证明 $\leftrightarrow$ 两个方向的有效性）（13 分）

$$F: \left(\exists x.\left(p(x) \rightarrow q(x)\right)\right) \leftrightarrow \left(\forall y.p(y) \rightarrow \exists z.q(z)\right)$$

➢ 参考答案：

F 是有效的；

1.$\exists x.(p(x) \rightarrow q(x)) \vdash \forall y.p(y) \rightarrow \exists z.q(z)$

$$\text{左存在}\dfrac{\text{右蕴含}\dfrac{\text{左蕴含}\dfrac{\text{左全称}\dfrac{\text{包含}\dfrac{}{p(c) \vdash p(c), \exists z.q(z)}}{\forall y.p(y) \vdash p(c), \exists z.q(z)} \quad \text{右存在}\dfrac{\text{包含}\dfrac{}{\forall y.p(y), q(c) \vdash q(c)}}{\forall y.p(y), q(c) \vdash \exists z.q(z)}}{p(c) \rightarrow q(c), \forall y.p(y) \vdash \exists z.q(z)}}{p(c) \rightarrow q(c) \vdash \forall y.p(y) \rightarrow \exists z.q(z)}}{\exists x.(p(x) \rightarrow q(x)) \vdash \forall y.p(y) \rightarrow \exists z.q(z)}$$

■

2.$\forall y.p(y) \rightarrow \exists z.q(z) \vdash \exists x.(p(x) \rightarrow q(x))$

$$\text{左蕴含}\dfrac{\text{右全称}\dfrac{\text{右存在}\dfrac{\text{右蕴含}\dfrac{\text{包含}\dfrac{}{p(a) \vdash p(a), q(a)}}{\vdash p(a), p(a) \rightarrow q(a)}}{\vdash p(a), \exists x.(p(x) \rightarrow q(x))}}{\vdash \forall y.p(y), \exists x.(p(x) \rightarrow q(x))} \quad \text{左存在}\dfrac{\text{右蕴含}\dfrac{\text{包含}\dfrac{}{p(c), q(c) \vdash q(c)}}{q(c) \vdash p(c) \rightarrow q(c)}}{\text{右存在}\dfrac{q(c) \vdash \exists x.(p(x) \rightarrow q(x))}{\exists z.q(z) \vdash \exists x.(p(x) \rightarrow q(x))}}}{\forall y.p(y) \rightarrow \exists z.q(z) \vdash \exists x.(p(x) \rightarrow q(x))}$$

■

8. 如下所示是二分查找算法 BinarySearch 的一种实现的源代码，其中|a|表示**整型数组** a 的长度，其它变量均默认为**整型**。BinarySearch 的前置条件和后置条件分别使用 **pre** 和 **post** 表示，其中的循环不变式使用 **inv** 表示。（其中分项列出的 **post** 以及 **inv** 将分别进行合取。）围绕如下 BinarySearch 实现的正确性验证，请回答以下问题。（14 分）

```
1   proc BinarySearch(a: array, key)
2     pre  ① sorted(a):
3     post ② (rv ≥ 0 → rv < |a| ∧ a[rv] = key)
4     post ③
5   {
6     rv := -1;
7     low := 0;
8     high := |a|;
9     while (rv = -1 ∧ low < high)
10      inv ④ inv1(low, high, |a|):
11      inv ⑤ rv ≥ 0 → rv < |a| ∧ a[rv] = key
```

```
12     inv ⑥ inv3(...):
13   {
14     mid := (low + high) / 2;
15     if (a[mid] < key) { low := mid + 1; }
16     else if (a[mid] > key) { high := mid; }
17     else { rv := mid; }
18   }
19   return rv;
20 }
```

1) 请在 **pre** ①位置，使用谓词逻辑公式形式化描述 BinarySearch 的前置条件 sorted(a)。它表示：数组 a 是**升序排列**的。（提示：使用全称量词）

① sorted(a): ∀ j,k. (0 ≤ j ≤ k < |a| → a[j] ≤ a[k])

2) 后置条件 **post** ②描述了 BinarySearch 实现的功能正确性的其中一种情况，即若返回值 rv 大于等于 0，则数组 a 的第 rv 个元素即为所查找到的 key。请在 **post** ③位置写出描述 BinarySearch 实现的功能正确性的**另一种情况**，即若返回值 rv 等于-1，则数组 a 中不存在值为 key 的元素。（提示：使用全称量词）

③ (rv < 0 → (∀ j. (0 ≤ j < |a| → a[j] ≠ key))

3) 为验证 BinarySearch 的后置条件 **post** 成立，需要分析的基本路径**共有____条**。请**额外**写出**至少两条**需要分析的基本路径所对应的验证条件（霍尔三元组）。以如下⑦所示的验证条件为例，其中，前置条件可以简写为 **pre**，后置条件可以简写为 **post**。此外，假定循环不变式已知，并可以简写为 **inv**。

⑦ {**pre**} rv = -1; low = 0; high = |a|; {**inv**}

⑧ _____

⑨ _____

共有 5 条

⑧ {**inv**} assume(rv=-1 ∧ low<high); mid:=(low+high)/2; assume(a[mid] < key); low:=mid+1; {**inv**}

⑨ {**inv**} assume(rv ≠ -1 ∨ low ≥ high); {**post**}

4) 注意到 BinarySearch 实现的第 15 行和第 16 行对数组 a 进行了**读操作**。请使用谓词逻辑公式，在 **inv** ④位置写出一个关于变量 low，high，以及|a|的循环不变式 inv1(low, high, |a|)，以证明数组的读操作不会发生越界。（提示：即证明访问数组时的下标都在正确的范围内。）

④ inv1(low, high, |a|): 0 ≤ low ≤ high ≤ |a|

5) 为了进一步验证 BinarySearch 实现的功能正确性，需要证明后置条件 **post** ②和 **post** ③都是成立的。举例来说，不变式 **inv** ⑤是一个符合要求的循环不变式，并且可以用于证明 **post** ②是成立的。请在 **inv** ⑥位置写出能够证明 **post** ③成立的循环不变式。

⑥ inv3(...): ∀ i.((0 ≤ i < low ∨ high ≤ i < |a|) → a[i] ≠ key)

9. 函数式编程是一种通过构建和应用函数来表达程序的编程范式。在函数式编程中，函数可以被作为其他函数的参数或者返回值。这里我们考虑一个简单的函数式语言：带常量扩展的 λ-演算。（17 分）

- **语法**

其语法定义如下:

变量: $x, y, z, \cdots$

常量: $c ::= 0 \mid 1 \mid 2 \mid \cdots$

项: $a, b, v ::= x \mid c \mid (\lambda x.\, a) \mid (a\ b)$

在这个语言中，核心的语法单元是“项”，其可以被视作一个表达式或一个程序。在上述定义中，项有四种形式。前两种是直接的，下面我们来解释一下后两种形式的含义。抽象 $(\lambda x.\, a)$ 的非形式化语义是说，我们定义一个匿名函数其参数为 x 并返回 a。我们称抽象 $(\lambda x.\, a)$ 在项 a 中绑定了变量 x。举例而言，$(\lambda x.\, 0)$ 的意思就是函数 $f(x) = 0$。应用 $(a\ b)$ 的非形式化语义是指在输入 b 上应用（调用）函数 a 的行为。比如说，$\big((\lambda x.\, x)\ 0\big)$ 的意思是去计算 $f(0)$，其中函数 $f(x) = x$。
为了避免引入不必要的语法惯例，在这个语言中，括号在形式语法中被直接定义，即是说括号不允许被省略。

定义（捕获避免的替换）：
我们把“将 a 中的 x 捕获避免的替换为 b”记作 $a[x \leftarrow b]$，其可以通过考虑 a 的形式来归纳定义。

(1) 如果 a 是一个常数 c，则 $a[x \leftarrow b]$ 为 c。
(2) 如果 a 就是变量 x，则 $a[x \leftarrow b]$ 为 b。
(3) 如果 a 是一个不同于 x 的变量 y，则 $a[x \leftarrow b]$ 为 y。
(4) 如果 a 是一个抽象 $(\lambda x.\, a')$，则 $a[x \leftarrow b]$ 为 $(\lambda x.\, a')$。
(5) 如果 a 是一个抽象 $(\lambda y.\, a')$，其中 x 和 y 是不同的变量，则 $a[x \leftarrow b]$ 是 $(\lambda z.\, a'[y \leftarrow z][x \leftarrow b])$，其中 z 对 b 来说是一个新的变量，也就是说，z 未在 b 中出现。
(6) 如果 a 是一个应用 $(a_1\ a_2)$，则 $a[x \leftarrow b]$ 为 $(a_1[x \leftarrow b]\ a_2[x \leftarrow b])$。

直观来说，捕获避免地替换就是说将 a 中的所有未被绑定的变量 x 换成 b。在这个过程中为了避免引入意外的绑定变量，我们需要改变一些绑定变量的选择，这种改变实际上并不会改变 a 的含义。

定义（值）：
如果 v 是常量 c 或者是抽象 $(\lambda x.\, b)$，则我们称 v 为一个值，记作 $v \in \text{Values}$。

- **自然语义**

这个语言一种标准的自然语义可以通过对下列推导规则的归纳应用来定义。它们定义了关系 $a \Rightarrow v$（直观的意思是“对 a 求值可以得到 v”）。

$$\frac{}{c \Rightarrow c}\,(\Rightarrow\text{-const}) \qquad \frac{}{(\lambda x.\, a) \Rightarrow (\lambda x.\, a)}\,(\Rightarrow\text{-fun})$$

$$\frac{a_1 \Rightarrow (\lambda x.\, b) \qquad a_2 \Rightarrow v_2 \qquad b[x \leftarrow v_2] \Rightarrow v}{(a_1\ a_2) \Rightarrow v}\,(\Rightarrow\text{-app})$$

示例（推导树）：下面是一棵 $(((\boldsymbol{\lambda x.\, x})\ (\boldsymbol{\lambda x.\, 0}))\ \boldsymbol{1}) \Rightarrow \boldsymbol{0}$ 的推导树，推导树需要包含注明推导的规则。

$$\frac{\dfrac{\dfrac{}{(\lambda x.\, x) \Rightarrow (\lambda x.\, x)}\,(\text{fun}) \quad \dfrac{}{(\lambda x.\, 0) \Rightarrow (\lambda x.\, 0)}\,(\text{fun})}{\big((\lambda x.\, x)\ (\lambda x.\, 0)\big) \Rightarrow (\lambda x.\, 0)}\,(\text{app}) \quad \dfrac{}{1 \Rightarrow 1}\,(\text{const}) \quad \dfrac{}{0 \Rightarrow 0}\,(\text{const})}{\Big(\big((\lambda x.\, x)\ (\lambda x.\, 0)\big)\ 1\Big) \Rightarrow 0}\,(\text{app})$$

示例（定理）：如果 $a \Rightarrow v$，则 $v \in \text{Values}$。
证明：我们在 $a \Rightarrow v$ 的推导树的高度 h 上作归纳。考虑这棵推导树最后一次应用的规则，即在树根节点处所应用的规则，有三种情况：

（⇒-const）这是归纳的基础情况。在这种情况中，v 是一个常量 c，所以根据定义即有 $v \in \text{Values}$。
（⇒-fun）这是归纳的基础情况。在这种情况中，v 是一个抽象 $(\lambda x.\, a')$，所以根据定义即有 $v \in \text{Values}$。
（⇒-app）当前考虑的规则的前提 $b[x \leftarrow v_2] \Rightarrow v$ 必然是一棵高度小于 h 的推导树的树根，根据归纳假设，我们即有 $v \in \text{Values}$。

- **结构操作语义**

这个语言一种标准的结构化操作语义是去定义一步规约关系 $\rightarrow$，这种关系可以通过下列规则来定义。

$$\frac{v \in \text{Values}}{((\lambda x.\, a)\ v) \rightarrow a[x \leftarrow v]}\ (\rightarrow\text{-}\beta)$$

$$\frac{a_1 \rightarrow a_2}{(a_1\ b) \rightarrow (a_2\ b)}\ (\rightarrow\text{-app-l}) \qquad \frac{a \in \text{Values} \qquad b_1 \rightarrow b_2}{(a\ b_1) \rightarrow (a\ b_2)}\ (\rightarrow\text{-app-r})$$

关系 $\overset{*}{\rightarrow}$（直观上意味着“a 可以通过零步、一步或多步规约到 b”）是关系 $\rightarrow$ 的自反传递闭包，可以被下列规则定义。

$$\frac{}{a \overset{*}{\rightarrow} a} \qquad \frac{a \rightarrow a' \qquad a' \overset{*}{\rightarrow} b}{a \overset{*}{\rightarrow} b}$$

示例（推导序列）： 下面是一个 $\left(((\lambda x.\, x)\ (\lambda x.\, 0))\ 1\right) \overset{*}{\rightarrow} 0$ 的推导序列。右侧所写的是在这一步规约中所使用到的规则。

$$\left(((\lambda x.\, x)\ (\lambda x.\, 0))\ 1\right)$$

$\rightarrow \left((\lambda x.\, 0)\ 1\right)$ $\quad$ $(\rightarrow\text{-app-l})$ and $(\rightarrow\text{-}\beta)$
$\rightarrow 0$ $\quad$ $(\rightarrow\text{-}\beta)$

请在仔细阅读上述对语法和语义的介绍之后，回答下面的问题。

1) 请通过构造一棵自然语义的推导树，来证明下面的陈述。

$$\left(\left((\lambda x.\, (\lambda y.\, (x\ y)))\ (\lambda x.\, x)\right)\ 1\right) \Rightarrow 1$$

$$\dfrac{\dfrac{\dfrac{}{(\lambda x.(\lambda y.(x\ y))) \Rightarrow (\lambda x.(\lambda y.(x\ y)))}\ (\text{fun}) \quad \dfrac{}{(\lambda x.x) \Rightarrow (\lambda x.x)}\ (\text{fun}) \quad \dfrac{}{(\lambda y.((\lambda x.x)\ y)) \Rightarrow (\lambda y.((\lambda x.x)\ y))}\ (\text{fun})}{((\lambda x.(\lambda y.(x\ y)))\ (\lambda x.x)) \Rightarrow (\lambda y.((\lambda x.x)\ y))}\ (\text{app}) \quad \dfrac{}{1 \Rightarrow 1}\ (\text{const})}{(((\lambda x.(\lambda y.(x\ y)))\ (\lambda x.x))\ 1) \Rightarrow 1}\ (\text{app}) \quad \dfrac{\dfrac{\dfrac{}{(\lambda x.x) \Rightarrow (\lambda x.x)}\ (\text{fun}) \quad \dfrac{}{1 \Rightarrow 1}\ (\text{const})}{((\lambda x.x)\ 1) \Rightarrow 1}\ (\text{app}) \quad \dfrac{}{1 \Rightarrow 1}\ (\text{const})}{}$$

2) 请通过构造一个结构操作语义的推导序列，来证明下面的陈述。

$$\left(\left((\lambda x.\, (\lambda y.\, (x\ y)))\ (\lambda x.\, x)\right)\ 1\right) \overset{*}{\rightarrow} 1$$

$$\left(\left((\lambda x.\, (\lambda y.\, (x\ y)))\ (\lambda x.\, x)\right)\ 1\right)$$

$\rightarrow \left(\lambda y.\, \left((\lambda x.\, x)\ \ y\right)\ \ 1\right)$ $\quad$ $(\rightarrow\text{-app-l})$ and $(\rightarrow\text{-}\beta)$
$\rightarrow \left((\lambda x.\, x)\ 1\right)$ $\quad$ $(\rightarrow\text{-}\beta)$
$\rightarrow 1$ $\quad$ $(\rightarrow\text{-}\beta)$

或

$$\left(\left((\lambda x.\, (\lambda y.\, (x\ y)))\ (\lambda x.\, x)\right)\ 1\right)$$

$\rightarrow \left(\lambda y.\, \left((\lambda x.\, x)\ \ y\right)\ \ 1\right)$ $\quad$ $(\rightarrow\text{-app-l})$ and $(\rightarrow\text{-}\beta)$
$\rightarrow \left((\lambda y.\, y)\ 1\right)$ $\quad$ $(\rightarrow\text{-app-l})$ and $(\rightarrow\text{-}\beta)$
$\rightarrow 1$ $\quad$ $(\rightarrow\text{-}\beta)$

3) 请证明下述定理。

如果 $a \Rightarrow v$，则 $a \xrightarrow{*} v$ 且 $v \in Values$。

我们在 $a \Rightarrow v$ 的推导树上作归纳。

考虑推导树的最后一次应用的规则，其有三种可能：

$\dfrac{}{c \Rightarrow c}\,(const)$：这是归纳的基础情况，在这种情况中，根据定义即得 $c \in Values$，而 $c \xrightarrow{*} c$ 可以通过规则 $\dfrac{}{c \xrightarrow{*} c}$ 直接证明。

$\dfrac{}{(\lambda x.b) \Rightarrow (\lambda x.b)}\,(fun)$：这是归纳的的基础情况，在这种情况中，根据定义即得 $(\lambda x.b) \in Values$，而 $c \xrightarrow{*} c$ 可以通过规则 $\dfrac{}{(\lambda x.b) \xrightarrow{*} (\lambda x.b)}$ 来直接证明。

$\dfrac{a_1 \Rightarrow (\lambda x.b) \qquad a_2 \Rightarrow v_2 \qquad b[x \leftarrow v_2] \Rightarrow v}{(a_1\ a_2) \Rightarrow v}\,(app)$:

由于 $a_1 \Rightarrow (\lambda x.b)$，根据归纳假设，$a_1 \xrightarrow{*} (\lambda x.b)$。所以，应用 $(\rightarrow$ -app-l)-规则若干次之后，我们便有 $(a_1\ a_2) \xrightarrow{*} ((\lambda x.b)\ a_2)$。

由于 $a_2 \Rightarrow v_2$，根据归纳假设，$a_2 \xrightarrow{*} v_2$ 且 $v_2 \in Values$。所以，应用 $(\rightarrow$ -app-r)-规则若干次之后，我们便有 $((\lambda x.b)\ a_2) \xrightarrow{*} ((\lambda x.b)\ v_2)$。

由于 $v_2 \in Values$，我们有 $\dfrac{v_2 \in Values}{((\lambda x.b)\ v_2) \rightarrow b[x \leftarrow v_2]}\,(\rightarrow -\beta)$。

因为 $b[x \leftarrow v_2] \Rightarrow v$，根据归纳假设，我们有 $b[x \leftarrow v_2] \xrightarrow{*} v$ 且 $v \in Values$.

因此，

$(a_1\ a_2)$

$\xrightarrow{*} ((\lambda x.b)\ a_2)$ 若干次 $(\rightarrow$ -app-l)-规则和归纳假设

$\xrightarrow{*} ((\lambda x.b) v_2)$ 若干次 $(\rightarrow$ -app-r)-规则和归纳假设

$\xrightarrow{*} b[x \leftarrow v_2]$ $(\rightarrow -\beta)$

$\xrightarrow{*} v$ 归纳假设

并且 $v \in Values$.