2022软件分析回忆版

选择题

1. 下列不是格的是

   A. S是有限集，$\langle \mathcal{P}(S), \subseteq \rangle$

   B. $\langle \mathbb{N}, \leqslant \rangle$

   C. $\langle \mathbb{N} \times \mathbb{N}, R \rangle$，其中$((a,b),(c,d)) \in R \Leftrightarrow a \leqslant c \wedge b \leqslant d$

   D. $\langle \{1,2,3,12,18,36\}, | \rangle$，其中$|$为整除关系

2. CHA分析下列代码，第6行`b.foo()`调用目标有

```
1  class A { void foo() {...} }
2  class B extend A { void foo() {...} }
3  class C extend B { void foo() {...} }
4  abstract class D extend B { void foo() {...} }
5
6  void bar(B b) { b.foo(); }
```

3. IR和AST相比，不具备的性质有

   A. 保留更多语法信息 B. 紧凑且接近机器代码 C. 语言无关　D. 有控制流信息

4. 格高度为H，CFG有N个节点，E条边，迭代求解器最多迭代次数为

   A. H+N　B. H×N　C. H+E　D. H×E

5. 下面关于PFG的说法不正确的是

   A. 有向图，表示对象的流动

   B. 无入边节点一定为变量，且为New语句的LValue

   C. 边起点的指针集一定为终点指针集的子集

   D. 边指针分析边构建

填空题

6. 对下面的代码进行CS指针分析，1-call-site时，`C.bar`的上下文有（），1-type时，`C.bar`的上下文有（）

```
1  static void main() {
2      B b1 = new B();
3      C c1 = new C();
4      b1.foo(c1);
5      B b2 = new B();
6      C c2 = new C();
7      b2.foo(c2);
8  }
9
10 class B {
11     void foo(C c) {
12         baz();
13         c.bar();
14     }
```

```
15      void baz() {
16          C x = new C();
17          x.bar();
18      }
19  }
20
21  class C { void bar() {...} }
```

7. 对下面的代码进行流敏感、上下文不敏感的常量传播，第5行的OUT为（），第7行的OUT为（），第11行的OUT为（）

```
1   static void main() {
2       int a = 2;
3       int b = 3;
4       int c = foo(a, b);
5       b = 5;
6       c = foo(a, b);
7       return;
8   }
9
10  static int foo(int x, int y) {
11      int z = x + y;
12      return z;
13  }
```

8. 本课程的英文名是（）
9. 填出污点分析中sink的规则，可以用Sinks表示sink方法和参数下标的集合

	Kind	Statement	Rule
Source	Call	`l: r = x.k(a1, ..., an)`	$\frac{l \to m \in CG, m \in Sources}{t_l \in pt(r)}$
Sink	Call	`l: r = x.k(a1, ..., an)`	

10. ICFG is （）s + （）&（）edges（第一空后的's'表复数）

简答题

11. may analysis，must analysis，over-approximated，under-approximated的意思和联系（如有）
12. 指针分析的key factors（写两个）以及它们的choices（写两个），指针分析和别名分析的关系
13. 填表

	Reaching Definitions	Live Variables	Available Expressions
Domain			
Direction			
May/Must			
Boundary			
Initialization			
Meet			

分析题

14. 找出下面代码中高密级到低密级的流，并说明是显示流还是隐式信道

```
h1 = l1 + 2;
if (l2 < 0) {
    l2 = h2 + 5;
} else if (h3 <= 0) {
    l3 = 0;
}
la[h4];
switch (h5) {
    case 1: h6 = 1;
    case 2: l4 = 2;
    default: l5 = 3;
}
```

15. 对下面的代码分别进行CI和2-object、1-heap的CS指针分析（填指针集），并结合调用图说明为什么CS比CI的精度更好

```
class Birthday {
    static void main() {
        Gift g1 = new Gift();
        Box b1 = new Box();
        Wrapper w1 = b1.wrapGift(g1);
        Gift r1 = w1.get();
        r1.info();
        Gift g2 = new SuperGift();
        Box b2 = new Box();
        Wrapper w2 = b2.wrapGift(g2);
        Gift r2 = w2.get();
        r2.info();
    }
}

class Box {
    Wrapper wrapGift(Gift e) {
        Wrapper w = new Wrapper();
        w.add(e);
        return w;
    }
}

```

```
class Wrapper {
    Gift gift;
    void add(Gift el) {
        this.gift = el;
    }
    Gift get() {
        Gift r = this.gift;
        return r;
    }
}

class Gift { void info() {...} }
class SuperGift extend Gift { void info() {...} }
```

16. 对下面的三地址码划分基本块（BB）并画出CFG

```
xxx
xxx
xxx
xxx
if xxx goto 10;
xxx
xxx
if xxx goto 5;
xxx
return;
```

代码题

17. 默写基于Worklist的Reaching Definitions算法的伪代码，开头已给出

```
OUT[entry] = ∅;
for (each basic block B\entry)
    OUT[B] = ∅;
```

18. 利用datalog实现Reaching Definitions算法（为简化，假设BB都只有1条指令），已知EDB有：

    - `Def(s, v, d)`，意为指令`s`是定义变量`v`的定义`d`
    - `Succ(s, ss)`，意为`ss`在CFG中是`s`的后继

    要生成的IDB有：

    - `Gen(s, d)`，意为指令`s` gen了定义`d`
    - `Kill(s, d)`，意为指令`s` kill了定义`d`
    - `In(s, d)`，意为`d`在`IN[s]`中
    - `Out(s, d)`，意为`d`在`OUT[s]`中

    给出`Gen`的代码作为示例：

```
Gen(s, d) <-
    Def(s, _, d)
```

    提示：可以用`s != s'`表示`s`和`s'`不是同一条指令

19. 如果一个表达式`x op y`在P点之后的每条执行流都在`x`或`y`被重定义前使用，那么我们称`x op y`在P点是very busy expression

如下面的代码中，`a + b`和`c + d`在P点是very busy expression，但`a - b`不是

```
P:
if (a + b > 0) {
    x = c + d;
    y = a - b;
} else {
    x = c + d;
    a = 0;
    y = a - b;
}
```

	Very Busy Expression
Domain	
Direction	
May/Must	
Boundary	
Initialization	
Transfer Function	
Meet	

Statement	**Gen**	**Kill**
`return 0;`	∅	∅
`x = 0; m = x + n;`		
`m = x + n; x = 0;`		