HOMEWORK ASSIGNMENT #3

CS589; Fall 2014

Due Date: **November 19, 2014**

Late homework 50% off

After **November 23** the homework assignment will not be accepted.

The **hardcopy** of the assignment must be submitted. Electronic submissions are not acceptable. Notice that the Blackboard homework assignment submissions are only considered as a proof of submission on time (before the deadline).

PROBLEM #1 (35 points): Testing polymorphism

For the following function *F()* and the inheritance relationships between five classes *side*, *A*, *B*, *C*, and *D*, design a set of test cases using **polymorphic testing**, i.e., each polymorphic call is "executed" at least once. For each test case show which polymorphic call(s) is executed. Notice that statements where polymorphic calls are made are highlighted in bold.

```
1:   int F(int a, int b, int c, int d){
     side *pa, *pb, *pc, *t;

2:        pa=new A;
3:        pb=new B;
4:        pa->set(a);
5:        pb->set(b);
6,7:      if (d>0) pc=new C;
8:        else pc=new D;
9:        pc->set(c);
10:       if (pa->get() > pb->get()) {
11:                 t = pa;
12:                 pa = pb;
13:                 pb = t;
          }
14:       if (pa->get() > pc->get()) {
15:                 t = pa;
16:                 pa = pc;
17:                 pc = t;
          }
18:       if (pb->get() > pc->get()) {
19:                 t = pb;
20:                 pb = pc;
21:                 pc = t;
          }
22:       if (pa->get() + pb->get() <= pc->get())
23:                 return 0;
24:       else return 1;
}
```

```
class side {
public:
virtual void set(int y) {x=y;};
virtual    void set_x(int y) {x=y;};
virtual    int get(){return x;};
private:
int x;
};

class A: public side {
public:
void set(int y) {if (y<0) set_x(1); else set_x(y);};
};

class B: public side {
public:
void set(int y) {if (y<10) set_x(10);else set_x(y);};
};

class C: public side {
public:
void set(int y) {if (y<5) set_x(5); else set_x(y);};
};

class D: public C {
public:
int get() {if (side::get()<10) return 0;
           else return side::get();}
};
```

A sample test case: Test #1: a=4, b=7, c=6, d=1; On this test function *F()* returns 0.

PROBLEM #2 (35 points): Symbolic evaluation

For the following function *triangle_type()* use symbolic evaluation to show that branch (20,21) is **not executable**. In your solution provide the **symbolic execution tree**.

```
1:      int triangle_type(int a, int b, int c)
{       int type, t;
2:      type=0;
3:      if (a > b) {
4:              t = a;
5:              a = b;
6:              b = t;
        }
7:      if (a > c) {
8:              t = a;
9:              a = c;
10:             c = t;
        }
11:     if (b > c) {
12:             t = b;
13:             b = c;
14:             c = t;
        }
15:     if (a + b <= c) {
16:             type = -1;
        }
        else {
17:             type = 3;
18:             if ((a == b) && (b == c)) {
19:                     type = 2;
                }
20:             else if (a == c)  {
21:                     type = 1;
                }
        }
22:     return type;
}
```

PROBLEM #3 (30 points): Program proving
Prove that the following function *F()* (that counts the number of absolute values of elements of array *a[]* that equal to the value of *x*) is correct with respect to the given pre-condition and post-condition:

Pre-condition: $1 \leq n \leq 100$
Post-condition:

$$s = \sum_{j=1}^{n} (|a[j]| == x)$$

Notice that **(|a[j]|==x)** is a Boolean expression that returns 1, if (|a[j]|==x) is true; otherwise, it returns 0.

```
1       int F (int a[], int n, int x) {
        int i, s, t;
2               i = n;
3               s = 0;
4               while (i > 0) {
5,6                     if (a[i]<0) t=-a[i];
7                       else t=a[i];
8,9                     if (t==x) s=s + 1;
10                      i = i - 1;
                };
11              return s;
        }
```