HOMEWORK ASSIGNMENT #3

CS589; Fall 2021
Due Date: **November 22, 2021**
Late homework 50% off
After **November 28** the homework assignment will not be accepted.

This is an **individual** assignment. **Identical or similar** solutions will be penalized.

Submission: All homework assignments must be submitted on the Blackboard as a **pdf file.** The hardcopy submissions will not be accepted.

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., for each polymorphic call all bindings should be "executed/tested" at least once. For each test case show which binding of the 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:          pc=new C;
4:          pa->set(a);
5:          pc->set(c);
6:          if (pa->get() < pc->get()) {
7:                t = pa;
8:                pa = pc;
9:                pc = t;
            }
10,11:      if (d<0) pb=new D;
12:         else pb=new B;
13:         pb->set(b);
14:         if (pa->get() > pc->get()) {
15:               t = pa;
16:               pa = pb;
17:               pb = t;
            }
18:         if (pa->get() > pb->get()) {
19:               t = pc;
20:               pc = pb;
21:               pb = t;
            }
22:         if (pa->get() + pc->get() <= pb->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<10) set_x(10); else set_x(y);};
};

class B: public side {
public:
void set(int y) {if (y<25) set_x(25);else set_x(y);};
};

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

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

A sample test case: Test #1: a=4, b=7, c=6, d=1

PROBLEM #2 (35 points): Symbolic evaluation
For the following function *F(int a, int b, int c)* use symbolic evaluation to show that the multiple-condition (True, False) in line 15 is **not executable**, i.e.,

((a == c) || (b == c))
True False

In your solution provide the **symbolic execution tree**.

```
1:      int F(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 == c) || (b == c)) {
16:                     type = 1;
                }
17:             return type;
        }
```

PROBLEM #3 (30 points): Program proving

The following function *F()* computes the summation of absolute values of elements of the array *a[]* consisting of *n* elements. Prove that function *F()* is correct for the given pre-condition and post-condition:

Pre-condition: $1 \le n \le 100$

Post-condition:

$$sum = \sum_{j=1}^{n} |a[j]|$$

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