HOMEWORK ASSIGNMENT #3

CS589; Fall 2022
Due Date: **November 16, 2022**
Late homework 50% off
After **November 21** 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. The submission **must** be as one PDF-file (otherwise, 10% penalty will be applied).

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

```
class side {
public:
side(){x=0; z=0;};
virtual    void set(int y, int a) {x=y; z=a;};
virtual    void set_x(int y) {x=y;};
virtual    void set_z(int y) {z=y;};
virtual    int get(){return x+z;};
private:
int x;
int z;
};

class A: public side {
public:
void set(int y, int a) {if (y<10) set_x(y); else set_z(a);};
};

class B: public side {
public:
void set(int y, int a) {if (y<5) set_x(y);else set_z(a);};
};

class C: public side {
public:
void set(int y, int a) {if (y>15) set_x(y); else set_z(a);};
};

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 *search* use symbolic evaluation to show that branch (10,11) is not executable (as a result, function *search* never returns value 1). In your solution provide the **symbolic execution tree.**

```
1       int search(int n, int x, int a[])
        {       int i, flag, z;
2               i = 1;
3               flag = -1;
4,5             if ((n>0)&&(a[i]==x)) flag = 1;
6               while ((i < n) && !(flag==1)) {
7,8                     if (a[i]==x) flag=1;
9                       i++;
                };
10,11           if ((n <=0) && (flag >= 1)) return 1;
12              else return 0;
        };
```

PROBLEM #3 (30 points): Program proving
The following function *F()* computes the the absolute minimum element in 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: **for all** $(1 \le j \le n)$: $(min \le |a[j]|)$

```
1   int abs_min(int a[], int n) {
      int i, min;
2       i=n-1;
3       min=a[n];
4,5     if (min<0) min=-min;
6       while (i > 0) {
7,8           if ((a[i]>=0)&&(min>a[i])) min = a[i];
9,10          else if ((a[i]<0)&&(min>-a[i])) min = -a[i];
11            i=i-1;
        }
12      return min;
13  }
```

For example, for the following input: a=(-10, 5, 7, -2, 3), n=5
The function returns: min = 2