CSci 4651 Fall 2003
Problem Set 9: Object-oriented languages I (Simula, Smalltalk, C++).
Due Friday, December 5th

Problem 1 (Simula object implementation). Consider the example in exercise 11.1 on p. 327. Consider replacing `cp.distance(r);` by each of the following lines of code:

- `r.equals(cp);`
- `cp.equals(cp);`

For each of the cases please draw the last activation record on the stack (the one that replaces activation record 4). Explain which method would be executed in each case (the one of the class Point or of the class ColorPt).

Problem 2 (Simula subtyping). Problem 11.3 p. 330, parts a and b only. Give a concrete example for part b.

Problem 3 (Smalltalk implementation). Exercise 11.4 on p. 330.

Problem 4 (Smalltalk inheritance). Exercise 11.7 on p. 333. In part (b), explain the two possible outcome and their consequences. Explain the trade-offs between the two. If you think that one of the outcomes is more consistent with the overall design of Smalltalk, please indicate which one (this question is optional).

If you would like to test C++ code, here is how to compile and run it on machines in the dungeon:

- write your code in a file with `.c` extension.
- in order to use `cout`, start your file with

  ```
  #include <iostream.h>
  ```

  If you are using `printf`, replace this line by

  ```
  #include <stdio.h>
  ```

  If you get an error message "library not found", please contact the dungeon masters (some C libraries were missing on some of the machines, they should be installed now).
- compile your program:

```
g++ myprog.c
```

- run your program by typing `a.out` at the prompt. If this doesn't work, try `./a.out`
- I have not tested C++ code given in the book. It's possible that some code fragments don't compile. Please contact me with any specific problems.

Note: Unless explicitly stated otherwise, exercises in this problem set don't require writing or testing programs.

Problem 5 (C++ objects). Exercise 12.1 on pp. 367-368.

Problem 6 (C++ inheritance). Consider the following declaration of classes A and B:

```
class A {
 protected:
  int x;
 public:
  A() { x = 0;}
  void increment() { x++; }
  virtual void decrement() { x--; }
  void print() {cout << x << endl;}
};

class B : public A {
 public:
  void increment() { x = x + 2; }
  void decrement() { x = x - 2; }
};
```

The following program is missing declarations of the types of pointers: they are replaced by question marks, each question mark stands for `A` or `B`.

```
void main() {
  A a;       // a is an object of class A
  ?1 * ptr;
  ptr = &a;
  ptr -> increment();
  ptr -> decrement();
  ptr -> print();
  B b;   // b is an object of class B
  ?2 * ptr1;
  ptr1 = &b;
  ptr1 -> increment();
  ptr1 -> decrement();
```

```
  ptr1 -> print();
}
```

For each of the results below please indicate whether it is a possible output of the program. If yes, what types do `?1` and `?2` stand for and which methods would be called? If no, please explain.

1. The program prints `0` and `0`.
2. The program prints `0` and `1`.
3. The program prints `0` and `-1`.
4. The program prints `-1` and `-1`.
5. The program prints `1` and `0`.

Problem 7 (C++ method subtyping and Thanksgiving cooking). Suppose that C++ class `Turkey` is a subclass of the class `Bird` and the class `TwentyPoundTurkey` is a subclass of `Turkey`. The class `Dish` has a method `cook` which takes a turkey and returns a nice Thanksgiving dish:

```
class Dish {
  public:
    virtual Dish cook(Turkey t) {
       Dish d = new Dish;
       .... // add some onions and carrots,
       .... // cook for a long time
       return d;
    }
    .... // other methods
};
```

Class `Casserole` is a subclass of class `Dish` which redefines the method `cook`. Below are 5 versions of `Casserole`:

1. ```
   class Casserole: public class Dish {
       public:
         virtual Casserole  cook(Turkey t) {
            Casserole c = new Casserole;
            .... // add some onions and carrots,
            .... // chop everything finely,
            .... // cook for a long time
            return c;
         }
    }
   ```
2. ```
   class Casserole: public class Dish {
       public:
   ```

```
    virtual Dish cook(Turkey t) {
       Dish d = new Dish;
       .... // add some onions and carrots,
       .... // chop everything finely,
       .... // cook for a long time
       return d;
    }
}
```

3.
```
class Casserole: public class Dish {
  public:
    virtual Casserole cook(Bird b) {
       Casserole c = new Casserole;
       .... // add some onions and carrots,
       .... // chop everything finely,
       .... // cook for a long time
       return c;
    }
}
```

4.
```
class Casserole: public class Dish {
  public:
    virtual Dish cook(Bird b) {
       Dish d = new Dish;
       .... // add some onions and carrots,
       .... // chop everything finely,
       .... // cook for a long time
       return d;
    }
}
```

5.
```
class Casserole: public class Dish {
  public:
    virtual Casserole cook(TwentyPoundTurkey t) {
       Casserole c = new Casserole;
       .... // add some onions and carrots,
       .... // chop everything finely,
       .... // cook for a long time
       return c;
    }
}
```

Question 1. For each redefinition of the method `cook` in the class `Casserole` say whether the function `main` below is valid if we replace the call to `Dish::cook` in `main` below by a call to the `Casserole::cook` without changing anything else in `main`. Note that it's OK if the program produces a different result after the method replacement, as long as the program doesn't have a compile or run-time error.

Please explain your answer in each case.

```
void main() {
   Turkey t = new Turkey();
   Dish d = Dish::cook(t);
   ...
}
```

Question 2. For each of the following redefinitions of `cook` in `Casserole` please indicate whether the new definition would be valid under the standard C++ inheritance rule, the permissive inheritance, and the contemporary inheritance.

Hint: Permissive inheritance corresponds to valid subtyping in question 1.

Problem 8 (C++ multiple inheritance). The code for this problem is available in my pub/4651 directory on epoxy in the file multiple.c.

You are given the following definition of classes `Color` and `Point`:

```
class Color {
 private:
  int red;
  int green;
  int blue;
 public:
  Color() {
    red = 128;
    green = 128;
    blue = 128;
  }
  virtual void print() {
    cout << "red = " << red << " green = ";
    cout << green << " blue = " << blue << endl;
  }
  virtual void greener(int x) {
    if (green + x < 256) green = green + x;
    else green = 256;
  }
};

class Point {
 private:
  int x;
  int y;
 public:
  Point() {
    x = 0;
    y = 0;
```

```
  }
  virtual void print() {
    cout << "x = " << x << " y = " << y << endl;
  }
  virtual void move (int xx, int yy) {
    x = x + xx;
    y = y + yy;
  }
};
```

Question 1. Write the class `ColorPoint` which inherits from both `Color` and `Point`. This class does not need a constructor (a new `ColorPoint` will be initialized by the default constructors of `Color` and `Point`). In the new class you need to overwrite the method `print` so that it calls the print method of `Color` to print the values of the colors and then calls the print method of `Point` to print out the coordinates. Do not add or overwrite any other methods.

Test your code on the following main function:

```
void main() {
  ColorPoint cp;
  cp.greener(50);
  cp.move(5, 4);
  cp.print();
}
```

Question 2. Draw the internal representation of a color point `cp`, including the values of the instance variables and the tables of virtual methods. Explain in detail how each of the methods and each private variable in the example above is found.