מבחן בתכנות מונחה עצמים בשפת C++
אוניברסיטת תל אביב

סמסטר קיץ – תשס"ה, מועד א'
תאריך הבחינה: 20.09.2005

מרצה: אוהד ברזילי

חומר עזר מותר בשימוש: כל חומר עזר כתוב ומשוכפל
(אסורים בשימוש: ספרים, מחשבים)

משך הבחינה: 3 שעות

הנחיות כלליות: המבחן כולל 7 עמודים (כולל עמוד זה)
4 שאלות ללא בחירה

יש לענות על כל השאלות במחברות המצורפות

המבחן מנוסח בלשון נקבה אולם הוא פונה לנשים וגברים כאחד

בהצלחה

שאלה מספר 1 (40 נקודות)

עבור קטעי הקוד הבאים רשמי מה יודפס כפלט:

א. (10 נקודות)

```
class Base1 {
public:
      Base1() { cout << "Base1::Base1()" << endl; }
      ~Base1() { cout << "Base1::~Base1()" << endl; }
};

class Derived1 : public Base1 {
public:
      Derived1() { cout << "Derived1::Derived1()" << endl; }
      ~Derived1() { cout << "Derived1::~Derived1()" << endl; }
};

class Base2 {
public:
      Base2() { cout << "Base2::Base2()" << endl; }
      ~Base2() { cout << "Base2::~Base2()" << endl; }
};

class Derived2 : public Base2 {
public:
      Derived2() { cout << "Derived2::Derived2()" << endl; }
      ~Derived2() { cout << "Derived2::~Derived2()" << endl; }
};

class Base {
      Base2 b2;
public:
      Base() { cout << "Base::Base()" << endl; }
      ~Base() { cout << "Base::~Base()" << endl; }
};

class Derived : public Base {
      Derived1 d1;
public:
      Derived() { cout << "Derived::Derived()" << endl; }
      ~Derived() { cout << "Derived::~Derived()" << endl; }
};
int main()
{
      Derived d;
      cout << "main()" << endl;
}
```

ב. (10 נקודות)

```
class Base1 {
public:
      Base1() { cout << "Base1::Base1()" << endl; }
      ~Base1() { cout << "Base1::~Base1()" << endl; }
};

class Derived1 : public Base1 {
public:
      Derived1() { cout << "Derived1::Derived1()" << endl; }
      ~Derived1() { cout << "Derived1::~Derived1()" << endl; }
};

class Base2 {
public:
      Base2() { cout << "Base2::Base2()" << endl; }
      ~Base2() { cout << "Base2::~Base2()" << endl; }
};

class Derived2 : public Base2 {
public:
      Derived2() { cout << "Derived2::Derived2()" << endl; }
      ~Derived2() { cout << "Derived2::~Derived2()" << endl; }
};

class Base {
      Base2 *b2;
public:
      Base() { cout << "Base::Base()" << endl; }
      ~Base() { cout << "Base::~Base()" << endl; }
};

class Derived : public Base {
      Derived1 d1;
public:
      Derived() { cout << "Derived::Derived()" << endl; }
      ~Derived() { cout << "Derived::~Derived()" << endl; }
};
int main()
{
      Base *b = new Derived();
      cout << "main()" << endl;
      delete b;
}
```

ג. (10 נקודות)

```
class Base1 {
public:
      Base1() { f(); }
      ~Base1() { f(); }
      void f() { cout << "Base1::f()" << endl; }
};

class Derived1 : public Base1 {
public:
      Derived1() { f(); }
      ~Derived1() { f(); }
      void f() { cout << "Derived1::f()" << endl; }
};

class Base2 {
public:
      Base2() { f(); }
      ~Base2() { f(); }
      void f() { cout << "Base2::f()" << endl; }
};

class Derived2 : public Base2 {
public:
      Derived2() { f(); }
      ~Derived2() { f(); }
      void f() { cout << "Derived2::f()" << endl; }
};

class Base {
      Base2 *b2;
public:
      Base() { f(); }
      ~Base() { f(); }
      void f() { cout << "Base::f()" << endl; }
};

class Derived : public Base {
      Derived1 d1;
public:
      Derived() { f(); }
      ~Derived() { f(); }
      void f() { cout << "Derived::f()" << endl; }
};

int main()
{
      Base *bd = new Derived();
      cout << "main()" << endl;
      bd->f();
      delete bd;
}
```

ד. (10 נקודות)

```
class Base1 {
public:
      Base1() { f(); }
      virtual ~Base1() { f(); }
      virtual void f() { cout << "Base1::f()" << endl; }
};

class Derived1 : public Base1 {
public:
      Derived1() { f(); }
      virtual ~Derived1() { f(); }
      virtual void f() { cout << "Derived1::f()" << endl; }
};

class Base2 {
public:
      Base2() { f(); }
      virtual ~Base2() { f(); }
      virtual void f() { cout << "Base2::f()" << endl; }
};

class Derived2 : public Base2 {
public:
      Derived2() { f(); }
      virtual ~Derived2() { f(); }
      virtual void f() { cout << "Derived2::f()" << endl; }
};

class Base {
      Base2 *b2;
public:
      Base() { f(); }
      virtual ~Base() { f(); }
      virtual void f() { cout << "Base::f()" << endl; }
};

class Derived : public Base {
      Derived1 d1;
public:
      Derived() { f(); }
      virtual ~Derived() { f(); }
      virtual void f() { cout << "Derived::f()" << endl; }
};
int main()
{
      Base *bd = new Derived();
      cout << "main()" << endl;
      bd->f();
      delete bd;
}
```

שאלה מספר 2 (20 נקודות)

נתונה המחלקה Date:

```
class Date {
      int d ,m , y ;
public:
      int day() const { return d ; }
      int month() const { return m ; }
      int year() const { return y ; }
      // ...
};
```

נרצה להוסיף למחלקה את המתודה `string_rep()` המחזירה את התאריך כמחרוזת (לצורכי הדפסה למשל).

א. האם מתודה זו היא שאילתה או פקודה? נמקי (2 נקודות)

ב. מהי החתימה של מתודה זו? (2 נקודות)

ג. לאחר עבודה עם המחלקה התגלה כי המתודה `string_rep()` לא יעילה מספיק, ויוצרת צוואר בקבוק בביצועי המחלקה. כדי לייעל את פעולת המחלקה נוסיף לה מנגנון של הטמנה (caching) כמתואר בקטע הקוד הבא:

```
class Date {
      mutable bool cache_valid;
      mutable string cache;
      void compute_cache_value() const;
      // ...
public:
      // ...
};
```

ממשי את `string_rep()` לפי החתימה מהסעיף הקודם. אם ברצונך לשנות את החתימה בעקבות הנתונים החדשים – נמקי זאת (8 נקודות).

ד. `cache` ו- `cache_valid` הוגדרו כ `mutable`. רישמי חוזה למתודה `string_rep()` שיסביר כי המתודה אינה משנה את המצב המופשט של `Date`. (8 נקודות).

שאלה מספר 3 (20 נקודות)

ביישום המטפל בצורות הנדסיות רבות (כגון: חרוט, פירמידה-משולשת, פירמידה-מרובעת, מלבן, ריבוע, משולש, מעגל ועוד...) נמצאו קטעי הקוד הבאים (חרוט == CONE):

```
// CONE.h

class CONE {
      double r;
      double h;
public:
      static const double PI;
      double volume() const;
      //...
};
```

```
// CONE.cpp

double const CONE::PI = 3.14159265358979323846264338327795;

double CONE::volume() const
{
      return (PI * r * r) * h / 3;
}
```

א. הסבירי במונחים של **הנדסת תוכנה** מדוע מימוש המתודה `volume` גרוע? אם יש יתרונות למימוש המוצג צייני גם אותם. (5 נקודות)

ב. תקני את המימוש. אם לצורך המימוש יש להגדיר מחלקות / מתודות / שדות נוספים הגדירי אותם. ניתן להשתמש בתעתיק הלועזי של מונחים באנגלית שאינך יודעת את פירושם (לדוגמא HAROOT במקום חרוט). אין צורך לממש אף מתודה פרט ל `volume()`. הסבירי בקצרה את הפתרון, אם השתמשת בשיטות כלליות של הנדסת תוכנה (כגון: תבנית עיצוב קלאסית) צייני את שם השיטה. (15 נקודות)

שאלה מספר 4 (20 נקודות)

א. בשפת C++ מותר להמיר Derived* ל- Base*. (למחלקה מותר להצביע בפועל לצאצאיה), ואולם ההמרה Derived** ל- Base** אסורה. מדוע? (5 נקודות)

ב. תני דוגמת קוד ב C++ המדגימה את ההשלכות הלא רצויות של המרה שכזו. (15 נקודות)

ושוב, בהצלחה