1. Разгледайте представения фрагмент от код и определете какви грешки има в него.
А) ред 6—не е коректно като резултат от функция да се връща стойността на локална променлива;
Б) няма проблеми;
В) ред 1—неправилно е използвана ключовата дума **enum**;
Г) ред 2—една функция не може да има тип на резултата `MY_TYPE`;

```
1 enum MY_TYPE {ONE, TWO, THREE};
2 MY_TYPE fun() {
3   MY_TYPE my_var;
4   my_var=ONE;
5   my_var=TWO;
6   return my_var;
7 }
```

---

2. Разгледайте представения фрагмент от код и определете какви са проблемите в него.
А) в ред 5—указателят `ptr` не може да се използва за промяна на стойността на обекта, към който сочи;
Б) в ред 4—неправилно е дефиниран указателят `ptr`;
В) в ред 6—не може да се променя стойността на указателят `ptr`;
Г) всичко е наред;

```
1 char* str1="Hello!";
2 char* str2="Hell!";
3 void fun(void) {
4   char const* ptr=str1;
5   ptr[2]='a';
6   ptr=str2;
7 }
```

---

3. Разгледайте представеният фрагмент от код и определете какви са проблемите в него.
А) ред 5—променливата `f` е предадена като константна препратка и поради това нейната стойност не може да се променя;
Б) ред 4—неправилно е дефинирана константна препратка към обект от класа `Foo`;
В) ред 5—член-променливата `f.x_` е недостъпна и не може да се използва;
Г) всичко е наред;

```
1 class Foo {
2   int x_;
3 //...
4   int bar(const Foo& f) {
5     return 1+f.x_;
6   }
7 };
```

---

4. Дадена е декларацията на класа `X`. В ред 8 трябва да се извика методът `f` на обекта, към който сочи указателя `X* xp`, обявен на ред 7. Кой е правилният начин да се направи това?
А) `xp->f;`
Б) `xp.f();`
В) `xp.f;`
Г) `xp->f();`

```
1 class X {
2   int v_;
3 public:
4   void f(void);
5 };
6 X x;
7 X* xp=&x;
8 //...
```

---

5. Разгледайте представения фрагмент от код. Определете какви са проблемите в него?
А) всичко е наред;
Б) променливата `size` е недостъпна в тялото на метода;
В) в ред 5 не може да се променя стойността на `size_`;
Г) за метода, дефиниран на ред 4, не е определен типът на резултата;

```
1 class X {
2   const int size_;
3 public:
4   X(void){
5   size_=128;
6   }
7 };
```

---

6. Дадена е декларацията на класа `X`. Указателят, дефиниран в ред 7 — `X* px` — е насочен към динамично създаден масив от 10 елемента от типа `X`. Кой е правилният начин да се освободи динамичната памет, към която сочи `px`?
А) **delete** `[ ] px;`
Б) **delete** `px;`
В) `free(px);`
Г) **delete** `px[ ];`

```
1 class X{
2 public:
3   X(void);
4   //...
5 };
6 //...
7 X* px;
```

7. Какви са проблемите в представеният код?
А) всичко е наред;
Б) в ред 8 член-променливата `x_` не е достъпна, тъй като е **private** променлива на базовият клас;
В) в производният клас не е деклариран конструктор;
Г) в ред 5 неправилно е използвана ключовата дума **public**;

```
1  class B {
2    int x_;
3  //...
4  };
5  class D: public B {
6  public:
7    void fun(void) {
8      x_*=10;
9    }
10 };
```

---

8. Кои от методите `get_a()` и `get_b()` могат да се извикват за обекта `d1`?
А) `d1.get_b()`;
Б) `d1.get_a()` и `d1.get_b()`;
В) `d1.get_a()`;
Г) нито един от двата метода;

```
1  class B {
2  public:
3    int get_a(void);
4  };
5  class D: public B {
6  public:
7    int get_b(void);
8  };
9  D d1;
```

---

9. Кои от член-променливите `x_` и `y_` на класа `B` са достъпни в методите на класа `D`?
А) `y_`;
Б) `x_` и `y_`;
В) `x_`;
Г) нито една от двете член променливи ;

```
1  class B {
2    int x_;
3  protected:
4    int y_;
5  //...
6  };
7  class D: public B {/*...*/};
```

---

10. В какъв ред се изикват деструкторите при унищожаване на обект от класа `D`?
А) зависи от вида наследяване;
Б) деструкторът на производния клас, деструкторът на базовия клас;
В) деструкторът на базовия клас, деструкторът на прозводния клас;
Г) в реда, описан от програмиста;

```
1  class B {
2    /*...*/
3  };
4  class D: public B {
5    /*...*/
6  };
```

---

11. Какви са проблемите в представения код?
А) в ред 3 — неправилно присвояване;
Б) в ред 5 — неправилна употреба на модификатора **public**;
В) в ред 10 — не могат да се създават обекти от класа `B`, тъй като той е абстрактен;
Г) в класа **class** `B` не е дефиниран конструктор по подразбиране;

```
1  class B {
2  public:
3    virtual void dump(void)const=0;
4  };
5  class D: public B {
6  public:
7    void dump(void)const
8    {/*...*/};
9  };
10 B* pb=new B;
11 pb->dump();
```

---

12. Кой от методите `dump()` ще се извика в ред 12 — `B::dump()` или `D::dump()`?

А) `D::dump()`, тъй като изборът между двата метода се извършва динамично, а действителният тип, към който сочи променливата `pb` е **class** `D`;

Б) `B::dump()`, тъй като изборът между двата метода се извършва динамично, а действителният тип, към който сочи променливата `pb` е **class** `B`;

В) `B::dump()`, тъй като изборът между двата метода се извършва статично;

Г) `D::dump()`, тъй като изборът между двата метода се извършва статично;

```
1  class B {
2  public:
3    void dump(void)const
4    {/*...*/};
5  };
6  class D: public B {
7  public:
8    void dump(void)const
9    {/*...*/};
10 };
11 B* pb=new D;
12 pb->dump();
```

---

13. Функцията `fun()`, дефинирана в ред 9 се извиква, както е показано в ред 12. Кой метод `dump()` ще се извика в тялото на функцията в ред 10 — `B::dump()` или `D::dump()`?

А) `B::dump()`, тъй като изборът на метод се извършва динамично, а действителният параметър на функцията е от типа `B`;

Б) `D::dump()`, тъй като изборът на метод се извършва динамично, а действителният параметър на функцията е от типа `D`;

В) `B::dump()`, тъй като в ред 9 се вика копиращ конструктор и в тялото на функцията се работи с копие на предадения параметър, което е от типа `B`;

Г) `D::dump()`, тъй като изборът на метод се извършва статично;

```
1  class B {
2  public:
3    virtual void dump(void)const;
4  };
5  class D: public B {
6  public:
7    void dump(void)const;
8  };
9  void fun(B b){
10   b.dump();
11 }
12 D d1; fun(d1);
```

---

14. Кой от методите `dump()` ще се извика в ред 12 — `B::dump()` или `D::dump()`?

А) `B::dump()`, тъй като изборът между двата метода се извършва динамично, а действителният тип, към който сочи променливата `pb` е **class** `B`;

Б) `D::dump()`, тъй като изборът между двата метода се извършва динамично, а действителният тип, към който сочи променливата `pb` е **class** `D`;

В) `D::dump()`, тъй като изборът между двата метода се извършва статично;

Г) `B::dump()`, тъй като изборът между двата метода се извършва статично;

```
1  class B {
2  public:
3    virtual void dump(void)const
4    {/*...*/};
5  };
6  class D: public B {
7  public:
8    void dump(void)const
9    {/*...*/};
10 };
11 B* pb=new D;
12 pb->dump();
```

---

1. В	5. Б	9. А	13. В
2. А	6. А	10. Б	14. Б
3. Г	7. Б	11. В	
4. Г	8. Б	12. В	