Вариант 1_2017 Ф.И.О._____№ группы_____

1	2	3	4	5	6	7	8	9	10

1. Вычеркните из функции main блоки, содержащие ошибки компиляции (предупреждения warning не считаем ошибкой). Что будет напечатано получившейся после вычеркивания программой? **Примечание**: считаем, что указатель можно без потерь преобразовать в long, а форма с многоточием поддерживает передачу любых объектов.

```
#include <iostream>
using namespace std;
struct A {
  A(int a=0){cout<<1;}
  A(const A&a){
      cout<<2;}
 ~A(){cout<<endl;}
};
```

```
struct B:  A {
  B(B& b){cout<<3;}
  B(const B& b){cout<<4;}
};

struct C:  B {
  C(...):B(*this){cout<<5;}
};
```

```
int main() {
   {A a((long)&a);}
   {A d=d=d;}
   {B b;}
   {const B b=b;}
   {C c(1); }
   {return 0;}
}
```

Ответ:

2. Измените данную программу, добавляя только операции разрешения области видимости :: так, чтобы было напечатано число 39.

```
#include <iostream>

using namespace std;
int x=0;
namespace X {  // int x=39;
    namespace Y { int x=17; }
    const int x=9;
}
```

```
struct A { int x;
    struct B{ int x; B(){ x=39;}} x1;
    A(){x=22;}
public:
    int h(){ { int x=39;}
        return     x +     x +     x; }
};
int main(){ A a; cout<< a.h(); return 0;}
```

3. Не изменяя оператор return и не используя символы /, \, ", %, #, добавьте всё необходимое в описание класса A так, чтобы компиляция прошла без ошибок.

```
class A { public:  int f(){      return (i,j)[i,j](i,j);}
};
```

Ответ:

4. Перечислите имена классов, описания которых содержат ошибки.

```
struct A {
  int register f(){
      return 1;}
  int const f(int x){
      return 2;}
};
```

```
struct B {
  static int  f(){
    return 1;}
  int  f() const{
    return 2;}
};
```

```
struct C {
  static int
  f(const int& x ){
    return 1;}
  int  f(int& x) {
    return 2;}
};
```

```
struct D {

  static virtual
  int f() {
    return 0 ;}
};
```

Ответ:

5. Что напечатает данная программа?

```
#include <iostream>
#include <typeinfo>
using namespace std;
class L { public: virtual void f (char x='L') {cout <<x<< "0"<<endl;}};
class P : public L { public: void f (char x='P') {cout <<x<< "1"<<endl;}};
int main() { P p; L * pl=&p; pl->f(); dynamic_cast<P*>(pl)->f(); return 0; }
```

Ответ:

6. Что будет напечатано в результате выполнения следующей программы?

```
#include <iostream>
using namespace std;
int f ( int n){
 try {  if (n<=0) throw n; f(--n); }
    catch(int& k){cout<<k++; throw; throw "b";}
    catch(const char *s){cout<<s;}
 throw "d";
}
```

```
int main() {
  try { cout<<'a'; f(2); cout<<'c';}
     catch (int k){cout<<k;}
     catch(const char *s){cout<<s;}
  cout<<'e';
  return 0;
}
```

Ответ:

7. Объясните ошибки компиляции для данной программы. Не исправляя имеющиеся строки, вставьте новую строку кода, в которой нет символов /, \, ″, %, #, (,), так, чтобы программа скомпилировалась. **Примечание**: угловые скобки использовать можно.

```
typedef int T;

T p( T a){ return a;}

float p( float a){ return a;}
```

```
int main() {
 return int(p('a')+p(3.14));
}
```

8. Может ли абстрактный класс в языке C++ являться абстрактным типом данных? Если да, то приведите пример, если нет – объясните почему. *Ответ:*

9. Добавьте в main() один оператор так, чтобы при выполнении программы произошла ошибка обращения по ссылке к уже не существующему объекту. Объясните причину ошибки. **Примечание:** считаем, что повторный вызов деструктора и обращение к членам объекта после его работы сами по себе не являются ошибками.

```
#include <iostream>
using namespace std;
class List { char elem; List & next;
public:
    List(): elem('\n'), next(*this){}
    List(const char*s): elem(*s? *s:'\n'), next(*s? * new List(s+1): *this){}
    List(const List& l):  elem(l.elem), next(&l!=&l.next? * new List(l.next): *this){}
    void print()const {cout<<elem; if(&next!=this) next.print();}
   ~List(){ if(&next!=this) {delete &next;}  }
};
int main() { List b("abc");           b.print(); }
```

10. Приведите функцию, эквивалентную данной, не содержащую новых конструкций из C++11.

```
#include <list>
int f(int x, const std::list<int>& l){
  auto p=l.begin();
  decltype(x) s=0;
  while(p!=l.end()) s+=*p++==x;
  return s;
}
```

Ответ:

Вариант 2_2017 Ф.И.О._____№ группы_____

1	2	3	4	5	6	7	8	9	10

1. Вычеркните из функции main блоки, содержащие ошибки компиляции (предупреждения warning не считаем ошибкой). Что будет напечатано получившейся после вычеркивания программой?
Примечание: считаем, что указатель можно без потерь преобразовать в long, а форма с многоточием поддерживает передачу любых объектов.

```
#include <iostream>
using namespace std;
struct X {
  X(int x=0){cout<<1;}
  X(const X&x){
      cout<<2;}
 ~X(){cout<<endl;}
};
```

```
struct Y:  X {
  Y(){cout<<3;}
  Y(Y& y):X(*this){
     cout<<4;}
};
struct Z:  Y {
  Z(...){cout<<5;}
};
```

```
int main() {
  {X x((long)&x);}
  {Y c(c=c);}
  {const X b=b;}
  {X p; Z q=p;}
  {Z z; }
  {return 0;}
}
```

Ответ:

2. Измените данную программу, добавляя только операции разрешения области видимости :: так, чтобы было напечатано число 27.

```
#include <iostream>
using namespace std;
int t=0;
namespace A {  // int t=27;
    namespace B { int t=14; }
    const int t=11;
}
```

```
struct X { int t;
    struct Y{ int t; Y(){ t=27;}} y1;
    X(){t=13;}
public:
    int h(){ { int t=27;}
        return     t +     t +     t; }
};
int main(){ X b; cout<< b.h(); return 0;}
```

3. Не изменяя оператор return и не используя символов /, \, ", %, #, добавьте всё необходимое в описание класса B так, чтобы компиляция прошла без ошибок.

```
class B { public:  int f(){      return b--->a<---b;} // пробелов в выражении нет
};
```

Ответ:

4. Перечислите имена классов, описания которых содержат ошибки.

```
struct W {
  int static
  g(const int& x ){
    return 1;}
  int const g(int&x){
    return 2;}
};
```

```
struct X {

  static int
  g()const{
    return 0;}
};
```

```
srtuct Y {
  static int g() {
    return 1;}
  virtual int
  g() const {
    return 2 ;}
};
```

```
struct Z {
  int register&g(){
      return 1; }
  int const
  g(int x){
    return 2;}
};
```

Ответ:

5. Что напечатает данная программа?

```
#include <iostream>
#include <typeinfo>
using namespace std;
class C { public: virtual void f (char x='C') {cout <<x<< "1"<<endl;}};
class D : public C { public: void f (char x='D') {cout <<x<< "2"<<endl;}};
int main(){ D d; C & rc=d; rc.f(); dynamic_cast<D&>(rc).f(); return 0; }
```

Ответ:

6. Что будет напечатано в результате выполнения следующей программы?

```
#include <iostream>
using namespace std;
int f ( int n){
 try{  if (n>0) throw n; throw "b"; }
   catch(int k){cout<<k--; f(k); }
   catch(const char *s){cout<<s; throw; }
}
```

```
int main() {
 try { cout<<'a'; f(2); cout<<'c';}
      catch (int k){cout<<k;}
      catch(const char *s){cout<<s;}
    cout<<'e';
 return 0;
}
```

Ответ:

7. Объясните ошибки компиляции для данной программы. Не исправляя имеющиеся строки, вставьте новую строку кода, в которой нет символов /, \, ″,%,#,(,), так, чтобы программа скомпилировалась. **Примечание**: угловые скобки использовать можно.

```
typedef int T;

short g( short  b){ return b;}

T g( T b){ return b;}
```

```
int main() {
 return int(g('b')+g(9.8));
}
```

8. Может ли некоторый класс в языке C++ являться абстрактным классом и при этом не быть абстрактным типом данных? Если да, то приведите пример, если нет – объясните почему. *Ответ:*

9. В процессе выполнения данной программы происходит ошибка обращения по «висячему» указателю, т.е. к уже не существующему объекту. Объясните причину ошибки. Не изменяя main() и представление данных, исправьте класс List так, чтобы этой ошибки не возникало. **Примечание:** считаем, что повторный вызов деструктора и обращение к членам объекта после его работы сами по себе не являются ошибками.

```
#include <iostream>
using namespace std;
class List { char elem; List *next;
public:
    List(): elem('\n'), next(this){}
    List(const char*s): elem(*s? *s:'\n'), next(*s? new List(s+1): this){}
    List(const List& l): elem(l.elem), next(&l!=l.next? new List(*l.next): this){}
    void print()const {cout<<elem; if(next!=this) next->print();}
   ~List(){ if(next!=this)  delete next; }
};
int main() { List c("cde");  c.~List();   c.print(); }
```

Ответ:

10. Приведите функцию, эквивалентную данной, не содержащую новых конструкций из C++11.

```
#include <vector>
int f(int y, const std::vector<int>& v){
   auto q=v.rbegin();
   decltype(y) s=0;
   while(q!=v.rend()) s+=*q++==y;
   return s;
}
```

Ответ:

Коллоквиум по С ++ 2017 (задачи, ответы и критерии)

А.А. Вылиток

За каждую задачу максимум 10 баллов. При вычитании баллов по критериям задачи полагаем y–x=0 при y < x.

Вариант 1

1. Вычеркните из функции main блоки, содержащие ошибки компиляции (предупреждения warning не считаем ошибкой). Что будет напечатано получившейся после вычеркивания программой? **Примечание**: считаем, что указатель можно без потерь преобразовать в long, а форма с многоточием поддерживает передачу любых объектов.

```
#include <iostream>
using namespace std;
struct A {
  A(int a=0){cout<<1;}
  A(const A&a){
      cout<<2;}
 ~A(){cout<<endl;}
};
```

```
struct B:  A {
  B(B& b){cout<<3;}
  B(const B& b){cout<<4;}
};

struct C:  B {
  C(...):B(*this){cout<<5;}
};
```

```
int main() {
   {A
 a((long)&a);}
   {A d=d=d;}
   {B b;}
   {const B b=b;}
   {C c(1); }
   {return 0;}
}
```

Ответ:

~~{B b;}~~
1
2
14
135

Область действия описываемого имени объекта начинается сразу после вхождения его описателя (описатели бывают вида p, *p, p( ) - функция, p[ ] - массив) в описание, перед инициализатором, если он есть (пункт 3.3.2 Стандарта 2011). В языках Си и С++ (в отличие от стандартного Паскаля) не считается ошибкой использование неинициализированного объекта (обращение к нему). То есть в выражении инициализации можно использовать описываемое имя объекта, например, для взятия адреса объекта.

[Обращение к полям (или методам) объекта или к его базовой части до начала выполнения конструктора или после окончания выполнения деструктора приводит (согласно пункту 12.7 Стандарта) к неопределенному поведению (undefined behavior) программы: например, в одной реализации такая программа может корректно выполниться (пункт 1.4.2 и сноска 2 в нем), в другой – приводить к ошибке компиляции или выполнения (см. 1.3.25)] [Передача потенциально-вычисляемого аргумента классового типа (9), имеющего нетривиальный конструктор копирования, нетривиальный конструктор перемещения или нетривиальный деструктор, в отсутствие соответствующего параметра, поддерживается условно; семантика такой передачи определяется реализацией (см. 5.2.2.7)]. [Компилятор Visual Studio не поддерживает форму инициализации B b (b);]

В первом блоке действует конструктор с целым параметром, напечатается 1.

При выходе из каждого блока прорабатывает деструктор базового класса, осуществляющий перевод строки.

Во втором блоке описание A d=d=d ; трактуется как A d(d=d), то есть сначала выполняется операция присваивания, сгенерированная автоматически, которая по сути ничего не делает, затем конструктор копирования, который тоже по сути ничего не копирует, но напечатает 2.

В третьем блоке ошибка, так как в классе B нет конструктора умолчания, это вычеркивается.

В четвертом блоке описание const B b=b; трактуется как const B b(b), то есть срабатывает константный конструктор копирования, который по сути ничего не копирует. Для базового класса вызывается конструктор умолчания, напечатается 1, затем конструктор производного класса напечатает число 4.

В пятом блоке для объекта `c` вызывается единственный конструктор «на все случаи жизни», который явно вызывает конструктор копирования (неконстантный) класса B, для которого вызывается конструктор умолчания класса A. В результате напечатается 135. (Конструктор может быть с многоточием, поскольку он является хотя и специальной, но функцией, а функция в С++ может быть описана с многоточием).

В шестом блоке лишь законный return, ошибки нет.

Критерии: -3 за каждое неверное вычеркивание (или невычеркивание) для всех блоков, кроме второго и четвертого; за каждую ошибочную напечатанную строку (в соответствии с невычеркнутыми блоками); за неверное разбиение на строки; -1 (всего) за вычеркивание второго и/или четвертого блока.

2. Измените данную программу, добавляя только операции разрешения области видимости :: так, чтобы было напечатано число 39.

```
#include <iostream>

using namespace std;
int x=0;
namespace X {  // int x=39;
    namespace Y { const int x=17; }
    const int x=9;
}
```

```
struct A { int x;
    struct B{ int x; B(){ x=39;}} x1;
    A(){x=22;}
public:
    int h(){ { int x=39;}
        return  x +   x +   x; }
};
int main(){ A a; cout<< a.h(); return 0;}
```

Ответ: нужно изменить выражение в операторе `return x + ::x + X::Y::x;`

Критерии: -4 за каждое неверное слагаемое (порядок слагаемых может быть любым, вместо x может быть A::x)

3. Не изменяя оператор return и не используя символов /, \, ″, %, #, добавьте всё необходимое в описание класса A так, чтобы компиляция прошла без ошибок.

```
class A { public:  int f(){      return (i,j)[i,j](i,j);}
};
```

Ответ:

```
class A { int a;
public:
   A(int a=0){this->a=a;}
   A operator[](int i){ return A(i);}
   A  operator ,(A a){return a;}
   int operator ()(A a1, A a2){ return a2.a+a1.a;}
     operator int(){return a;}
   int f(){A i,j; return (i,j)[i,j](i,j);}

};
```

Возможны вариации решения. Запрет использования / и ″ не позволяет закомментировать фрагмент или сделать его строкой, запрет # не позволяет макросы, запрет \ – эскейп-последовательности и переносы в макросах, запрет % не позволяет использовать дограф %: как альтернативу для #. [Триграфы исключены из Стандарта 2017].

Критерии: -5 за неверную трактовку выражения, -2 за каждую ошибку при описании класса

4. Перечислите имена классов, описания которых содержат ошибки.

```
struct A {
  int register f(){
      return 1;}
  int const f(int x){
      return 2;}
};
```

```
struct B {
  static int  f(){
    return 1;}
  int  f() const{
    return 2;}
};
```

```
struct C { public:
  static int
  f(const int& x ){
    return 1;}
  int  f(int& x) {
    return 2;}
};
```

```
struct D {

  static virtual
  int f() {
    return 0 ;}
};
```

Ответ: A, B, D.

В классе A ошибочно использован спецификатор register – он применяется только к данным. В классе B статическая функция не может быть перегружена с нестатической функцией с такими же параметрами. В классе C ошибок нет, нужная функция выбирается в зависимости от константности фактического параметра. В классе D ошибка в том, что статическая функция не получает скрытый параметр this на объект, от имени которого она вызвана, и поэтому не может непосредственно обращаться к нестатическим полям своего объекта (в т.ч. к скрытому полю pvtbl, содержащему адрес ТВМ) и, следовательно, не может быть виртуальной.

Критерии: -3 за каждый пропущенный или лишний класс.

5. Что напечатает данная программа?

```
#include <iostream>
#include <typeinfo>
using namespace std;
class L { public: virtual void f (char x='L') {cout <<x<< "0"<<endl;}};
class P : public L { public: void f (char x='P') {cout <<x<< "1"<<endl;}};
int main() { P p; L * pl=&p; pl->f(); dynamic_cast<P*>(pl)->f(); return 0; }
```

Ответ: L1
P1

Параметр по умолчанию для функции, вызванной через указатель или ссылку, определяется статически (в момент компиляции) по типу указателя или ссылки.

Критерии: -5 за каждую неверную строку.

6. Что будет напечатано в результате выполнения следующей программы?

```
#include <iostream>
using namespace std;
int f ( int n){
 try {  if (n<=0) throw n; f(--n); }
    catch(int& k){cout<<k++; throw; throw "b";}
    catch(const char *s){cout<<s;}
 throw "d";
}
```

```
int main() {
  try { cout<<'a'; f(2); cout<<'c';}
     catch (int k){cout<<k;}
     catch(const char *s){cout<<s;}
  cout<<'e';
  return 0;
}
```

Ответ: `a0123e` (исключение перебрасывается в предыдущий рекурсивный вызов с увеличением на 1).

Критерии: -3 за каждый неверно напечатанный (или пропущенный) символ.

7. Объясните ошибки компиляции для данной программы. Не исправляя имеющиеся строки, вставьте новую строку кода, в которой нет символов `/`, `\`, `"`, `%`, `#`, `(`, `)`, так, чтобы программа скомпилировалась.

```
typedef int T;

T p( T a){ return a;}

float p( float a){ return a;}
```

```
int main() {
 return int(p('a')+p(3.14));
}
```

Ответ: ошибка в неоднозначности для вызова p(3.14): на шаге (в) алгоритма best-matching подходят обе функции p(). Нужно вставить строку `template<class T>` непосредственно перед `T p( T a){ return a;}` Запрет круглых скобок не позволяет добавить перегруженную без шаблона функцию.

Критерии: -5, если неверно объяснена ошибка (не найдена, найдены лишние)
-5, если неверно вставлена строка (или не вставлена)

8. Может ли абстрактный класс в языке C++ являться абстрактным типом данных? Если да, то приведите пример, если нет – объясните почему.

Ответ: В C++ АТД реализуется с помощью классов (структур), в которых нет открытых членов-данных. Абстрактный класс содержит чистую виртуальную функцию. Следовательно, ответ положительный, вот пример абстрактного класса, являющегося АТД:

```
class A { public: virtual void f()=0;}
```

Критерии: -10 за неверный (отсутствующий) ответ, ошибочные примеры или объяснения.

9. Добавьте в main() один оператор так, чтобы в процессе выполнения программы произошла ошибка обращения по ссылке к уже не существующему объекту. Объясните причину ошибки. **Примечание:** считаем, что повторный вызов деструктора и обращение к членам объекта после его работы сами по себе не являются ошибками.

```
#include <iostream>
using namespace std;
class List { char elem; List & next;
public:
    List(): elem('\n'), next(*this){}
    List(const char*s): elem(*s? *s:'\n'), next(*s? * new List(s+1): *this){}
    List(const List& l):  elem(l.elem), next(&l!=&l.next? * new List(l.next): *this){}
    void print()const {cout<<elem; if(&next!=this) next.print();}
   ~List(){ if(&next!=this) {delete &next;}  }
};
int main() { List b("abc");           b.print(); }
```

Ответ:

```
int main() { List b("abc");  b.~List(); b.print();  }
```

При явном вызове деструктора освобождается память, занимаемая звеньями списка, причем значение поля next объекта b становится неопределенным, но при следующем вызове `b.print()` делается попытка его использования. [Повторный вызов деструктора относится к неопределенному поведению (12.4.15). После окончания работы деструктора, обращение к членам объекта также относится к неопределённому поведению (12.7.1). Это означает широкий диапазон поведения в разных реализациях: в одной реализации такая программа считается корректной (что предполагалось в условии задачи), в другой может фиксироваться ошибка, так как объект «перестает существовать» с момента первого вызова деструктора.]

Критерии: -5 если не предложено верное добавление
-5 если нет правильного объяснения

10. Приведите функцию, эквивалентную данной, не содержащую новых конструкций из C++11.

```
#include <list>
int f(int x, const std::list<int>& l){
  auto p=l.begin();
  decltype(x) s=0;
  while(p!=l.end()) s+=*p++==x;
  return s;
}
```

Ответ:

```
#include <list>
int f(int x, const std::list<int>& l){
  std::list<int>::const_iterator p=l.begin();
  int s=0;
  while(p!=l.end()) s+=*p++==x;
  return s;
}
```

Критерии: - 5 за каждую неверно замененную конструкцию (auto, decltype).

Вариант 2

1. Вычеркните из функции main блоки, содержащие ошибки компиляции (предупреждения warnig не считаем ошибкой). Что будет напечатано получившейся после вычеркивания программой? **Примечание**: считаем, что указатель можно без потерь преобразовать в long, а форма с многоточием поддерживает передачу любых объектов.

```
#include <iostream>
using namespace std;
struct X {
  X(int x=0){cout<<1;}
  X(const X&x){
      cout<<2;}
 ~X(){cout<<endl;}
};
```

```
struct Y:  X {
  Y(){cout<<3;}
  Y(Y& y):X(*this){
     cout<<4;}
};
struct Z:  Y {
  Z(...){cout<<5;}
};
```

```
int main() {
  {X x((long)&x);}
  {Y c(c=c);}
  {const X b=b;}
  {X p; Z q=p;}
  {Z z; }
  {return 0;}
}
```

Ответ:

~~{X p; Z q=p;}~~
1
24
2
135

Область действия описываемого имени объекта начинается сразу после вхождения его описателя (описатели бывают вида p, *p, p( ) - функция, p[ ] - массив) в описание, перед инициализатором, если он есть (пункт 3.3.2 Стандарта 2011). В языках Си и С++ (в отличие от стандартного Паскаля) не считается ошибкой использование неинициализированного объекта (обращение к нему). То есть в выражении инициализации можно использовать описываемое имя объекта, например, для взятия адреса объекта. [Обращение к полям (или методам) объекта или к его базовой части до начала выполнения конструктора или после окончания выполнения деструктора приводит (согласно пункту 12.7 Стандарта) к неопределенному поведению (undefined behavior) программы: например, в одной реализации такая программа может корректно выполниться (пункт 1.4.2 и сноска 2 в нем), в другой – приводить к ошибке компиляции или выполнения (см. 1.3.25)].[Передача потенциально-вычисляемого аргумента классового типа (9), имеющего нетривиальный конструктор копирования, нетривиальный конструктор перемещения или нетривиальный деструктор, в отсутствие соответствующего параметра, поддерживается условно; семантика такой передачи определяется реализацией (см. 5.2.2.7)]. [Компилятор Visual Studio не поддерживает форму инициализации X b (b);]

В первом блоке действует конструктор с целым параметром, напечатается 1. Согласно пункту 5.2.10.4 Стандарта 2011, указатель может быть преобразован к целому типу, достаточному для его представления.

При выходе из каждого блока прорабатывает деструктор базового класса, осуществляющий перевод строки.

Во втором блоке сначала выполняется операция присваивания (для вычисления выражения в скобках), сгенерированная автоматически, которая по сути ничего не делает, затем конструктор копирования, который тоже по сути ничего не копирует, сначала он явно вызывает конструктор базового класса, который напечатает 2, после чего конструктор производного класса напечатает 4.

В третьем блоке описание const X b=b; трактуется как const X b(b), то есть срабатывает конструктор копирования, который по сути ничего не копирует. Напечатается 2.

В четвертом блоке ошибка, так как делается попытка инициализировать объект производного класса объектом базового класса, это вычеркивается.

В пятом блоке для объекта z вызывается единственный конструктор «на все случаи жизни», вызывающий конструктор умолчания класса B, для которого вызывается конструктор умолчания класса A. В результате напечатается 135. (Конструктор может быть с многоточием, поскольку он является хотя и специальной, но функцией, а функция в С++ может быть описана с многоточием).

В шестом блоке лишь законный return, ошибки нет.

Критерии: -3 за каждое неверное вычеркивание (или невычеркивание) для всех блоков, кроме второго и третьего; за каждую ошибочную напечатанную строку (в соответствии с невычеркнутыми блоками); за неверное разбиение на строки; -1 (всего) за вычеркивание второго и/или третьего блока.

2. Измените данную программу, добавляя только операции разрешения области видимости :: так, чтобы было напечатано число 27.

```
#include <iostream>

using namespace std;
int t=0;
namespace A {  // int t=27;
    namespace B { const int t=14; }
    const int t=11;
}
```

```
struct X { int t;
    struct Y{ int t; Y(){ t=27;}} y1;
    X(){t=13;}
public:
    int h(){ { int t=27;}
        return  t +   t +    t; }
};
int main(){ X b; cout<< b.h(); return 0;}
```

Ответ: нужно изменить выражение в операторе `return  t + ::t + A::B::x;`

Критерии: -4 за каждое неверное слагаемое (порядок слагаемых может быть любым, вместо t может быть X::t)

3. Не изменяя оператор return и не используя символов /, \, ″, %, #, добавьте всё необходимое в описание класса B так, чтобы компиляция прошла без ошибок.

```
class B { public:  int f(){      return b--->a<---b;} // пробелов в выражении нет
};
```

Ответ: Если *b* описать как объект класса *B*, содержащего поле *a,* то выражение можно трактовать следующим образом: `b.operator--(int).operator->() < b.operator-(). operator--().operator int()`. Остается определить в классе необходимые операции и, возможно, конструкторы.

```
class B { int a;
public:
   B(int a=0){this->a=a;}
   B & operator--(){a--; return *this;}
   B operator--(int){B temp(*this);a--; return temp;}
   B * operator->(){return this;}
   B operator -(){B temp(-a); return temp;}
     operator int(){return a;}
   int f(){B b;  return b--->a<---b;}
};
```

Возможны вариации решения. Запрет использования / и ″ не позволяет закомментировать фрагмент или сделать его строкой, запрет # не позволяет макросы, запрет \ – эскейп-последовательности и переносы в макросах, запрет % не позволяет использовать дограф `%:` как альтернативу для #. [Триграфы исключены из Стандарта 2017].

Критерии: -5 за неверную трактовку выражения, -2 за каждую ошибку при описании класса.

4. Перечислите имена классов, описания которых содержат ошибки.

```
struct W {
  int static
  g(const int& x ){
    return 1;}
  int const g(int&x){
    return 2;}
};
```

```
struct X {

  static int
  g()const{
    return 0;}
};
```

```
struct Y {
  static int g() {
    return 1;}
  virtual int
  g() const {
    return 2 ;}
};
```

```
struct Z {
  int register&g(){
      return 1; }
  int const
  g(int x){
    return 2;}
};
```

Ответ: X, Y, Z.

В классе W ошибок нет, нужная функция выбирается в зависимости от константности фактического параметра. В классе X ошибка в том, что статическая функция по определению не пользуется нестатическими полями (в т.ч. воображаемым полем this) и, следовательно, не может быть константной. В классе Y статическая функция не может быть перегружена с нестатической функцией с такими же параметрами. В классе Z ошибочно использован спецификатор register – он применяется только к данным в блоке или параметрам функции (7.1.1.2). [В Стандарте 2017 http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4659.pdf слово register всё ещё остаётся зарезервированным, но уже не означает ничего]

Критерии: -3 за каждый пропущенный или лишний класс.

5. Что напечатает данная программа?

```
#include <iostream>
#include <typeinfo>
using namespace std;
class C { public: virtual void f (char x='C') {cout <<x<< "1"<<endl;}};
class D : public C { public: void f (char x='D') {cout <<x<< "2"<<endl;}};
int main(){ D d; C & rc=d; rc.f(); dynamic_cast<D&>(rc).f(); return 0; }
```

Ответ: C2
D2
Параметр по умолчанию для функции, вызванной через указатель или ссылку, определяется статически (в момент компиляции) по типу указателя или ссылки.

Критерии: -5 за каждую неверную строку.

6. Что будет напечатано в результате выполнения следующей программы?

```
#include <iostream>
using namespace std;
int f ( int n){
 try{  if (n>0) throw n; throw "b"; }
   catch(int k){cout<<k--; f(k); }
   catch(const char *s){cout<<s; throw; }
}
```

```
int main() {
 try { cout<<'a'; f(2); cout<<'c';}
      catch (int k){cout<<k;}
      catch(const char *s){cout<<s;}
    cout<<'e';
 return 0;
}
```

Ответ: `a21bbe` (функция рекурсивно вызывается из обработчика).

Критерии: -3 за каждый неверно напечатанный (или пропущенный) символ.

7. Объясните ошибки компиляции для данной программы. Не исправляя имеющиеся строки, вставьте новую строку кода, в которой нет символов `/`, `\`, `"`, `%`, `#`, `(`, `)`, так, чтобы программа скомпилировалась.

```
typedef int T;

short g( short  b){ return b;}

T g( T b){ return b;}
```

```
int main() {
 return int(g('b')+g(9.8));
}
```

Ответ: ошибка в неоднозначности для вызова g(9.8): на шаге (в) алгоритма best-matching подходят обе функции g(). Нужно вставить строку `template<class T>` непосредственно перед `T g( T b){ return b;}`. Запрет круглых скобок не позволяет добавить перегруженную без шаблона функцию.

Критерии: -5, если неверно объяснена ошибка (не найдена, найдены лишние).
-5, если неверно вставлена строка (или не вставлена).

8. Может ли некоторый класс в языке C++ являться абстрактным классом и при этом **не** быть абстрактным типом данных? Если да, то приведите пример, если нет – объясните почему.

Ответ: В C++ АТД реализуется с помощью классов (структур), в которых нет открытых членов-данных. Абстрактный класс содержит чистую виртуальную функцию. Следовательно, ответ положительный, вот пример абстрактного класса, не являющегося АТД:

```
class A { public: int a; virtual void f()=0;}
```

Критерии: -10 за неверный (отсутствующий) ответ, ошибочные примеры или объяснения.

9. В процессе выполнения данной программы происходит ошибка обращения по «висячему» указателю, т.е. к уже не существующему объекту. Объясните причину ошибки. Не изменяя main() и представление данных, исправьте класс List так, чтобы этой ошибки не возникало. **Примечание:** считаем, что повторный вызов деструктора и обращение к членам объекта после его работы сами по себе не являются ошибками.

```
#include <iostream>
using namespace std;
class List { char elem; List *next;
public:
    List(): elem('\n'), next(this){}
    List(const char*s): elem(*s? *s:'\n'), next(*s? new List(s+1): this){}
    List(const List& l): elem(l.elem), next(&l!=l.next? new List(*l.next): this){}
    void print()const {cout<<elem; if(next!=this) next->print();}
   ~List(){ if(next!=this)  delete next; }
};
int main() { List c("cde");  c.~List();   c.print(); }
```

Ответ: При явном вызове деструктора освобождается память, занимаемая звеньями списка, причем значение поля next объекта `c` становится неопределенным, но при следующем вызове `c.print()` делается попытка его использования.

Достаточно исправить деструктор следующим образом:

```
~List(){ if(next!=this) { delete next; next=this; elem='\n';}    }
```

[Повторный вызов деструктора относится к неопределенному поведению (12.4.15). После окончания работы деструктора, обращение к членам объекта также относится к неопределённому поведению (12.7.1). Это означает широкий диапазон поведения в разных реализациях: в одной реализации такая программа считается корректной (что предполагалось в условии задачи), в другой может фиксироваться ошибка, так как объект «перестает существовать» с момента первого вызова деструктора.]

Критерии: - 5 если ошибка не найдена или не объяснена правильно,
- 5 если не предложено верное исправление.

10. Приведите функцию, эквивалентную данной, не содержащую новых конструкций из C++11.

```
#include <vector>
int f(int y, const std::vector<int>& v){
  auto q=v.rbegin();
  decltype(y) s=0;
  while(q!=v.rend()) s+=*q++==y;
  return s;
}
```

Ответ:

```
#include <vector>
int f(int y, const std::vector<int>& v){
  std::vector<int>::const_reverse_iterator q=v.rbegin();
  int s=0;
  while(q!=v.rend()) s+=*q++==y;
  return s;
}
```

Критерии: - 5 за каждую неверно замененную конструкцию (auto, decltype).