Вариант 1_2019 Ф.И.О._____№ группы_____

1	2	3	4	5	6	7	8	9	10

1. Ответьте на следующие вопросы. Обоснуйте ответ на каждый вопрос.

(а) Может ли грамматика одновременно быть регулярной и не быть контекстно-зависимой?

(б) Может ли конечный язык быть неоднозначным?

(в) Существует ли регулярный язык, который не порождается неукорачивающей грамматикой?

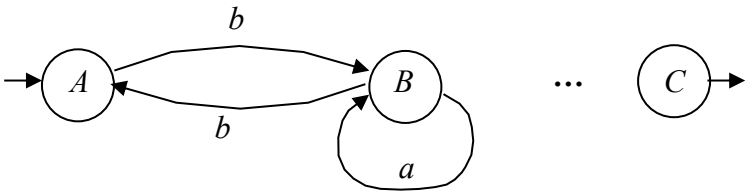
2. Постройте грамматику выражений над переменными a и b с операциями сложения и умножения и унарным минусом (наивысший приоритет), к которой применим метод рекурсивного спуска. Добавьте в грамматику действия только вида ⟨ *cout* <<'символ'; ⟩, реализующие формальный перевод $\tau = \{ (x, y) \mid$ x – арифметическое выражение, y – эквивалентное арифметическое выражение, не содержащее двух рядом стоящих унарных минусов $\}$. Пример: $-(a + --b * ----(a))$ ➔ $-(a + b * -(a))$.

3. Даны названия программных систем: *CVS, SVN, GIT, Make*. Вычеркнуть среди них одно лишнее и дать определение понятию, примерами которого являются все оставшиеся системы.

Ответ:

4. Даны: 1) часть множества P_L леволинейной грамматики $G_L = \langle \{A, B, C\}, \{a, b\}, P_L, C \rangle$; 2) часть множества P_R праволинейной грамматики $G_R = \langle \{A, B, C\}, \{a, b\}, P_R, A \rangle$; 3) часть множества дуг δ конечного автомата $A = \langle \{A, B, C\}, \{a, b\}, \delta, \{A\}, \{C\} \rangle$.

δ:



P_L: $A \to Aa \mid \varepsilon$

...

P_R: $C \to aC \mid \varepsilon$

...

Пополнить δ, P_L, P_R, (дописать недостающие правила, дорисовать дуги) так, чтобы одновременно выполнялись условия: (а) грамматики G_L и G_R приведённые и эквивалентные; (б) A получается из G_L алгоритмом построения конечного автомата по леволинейной грамматике; (в) A получается из G_R алгоритмом построения конечного автомата по праволинейной грамматике; (г) A не является детерминированным; (д) количество дуг в A не превосходит 6.

5. Дана польская запись фрагмента программы на Си. Восстановите этот фрагмент на языке Си без операторов перехода. Приведите еще один (отличающийся, но также без операторов перехода) текстовый фрагмент, имеющий такую же польскую запись. **Примечание:** пустой оператор не занимает место в польской записи.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ПОЛИЗ	&b	+#	7	!F	7	!	&a	–#	;	a	b	+	17	!F	1	!	

Ответ:

6. Если в функции main есть блоки, содержащие ошибки компиляции, вычеркните их. Что будет напечатано получившейся программой? Считать, что компилятор ничего не оптимизирует.

```
#include <iostream>
using namespace std;
struct A {
  A(int a){cout<<1;}
  A(const A&a = A(1)){
      cout<<2;}
  A(A& a){cout<<3;}
 ~A(){cout<<'a';}
};
```

```
struct B:  A {

 B(const B& b){ cout<<4;}

 B(int i=0):A(*this){
                cout<<5;}
 ~B(){cout<<'b';}
};
```

```
int main(){
 {cout<< "start";}
 {cout<<endl; A a(1);}
 {cout<<endl; A a;}
 {cout<<endl; B b;}
  return 0;
}
```

Ответ:

7. (а) Привести описания не более двух функций, чтобы были верны следующие обращения:

```
g(50L);  g((const double) 1); g ();   g ("str"); g (0, 0);   g (3, "str");
```

(б) Может ли константный член класса быть статическим ? Обоснуйте ответ.

8. Может ли в выражении вида *a+b* проявиться одновременно и статический, и динамический полиморфизм? Обоснуйте ответ. Какой еще вид полиморфизма есть в C++?

9. Что будет напечатано в результате работы данной программы? В курсе рассматривались стандартные исключения bad_alloc, bad_cast, bad_typeid.

```
#include <iostream>
#include <exception>
#include <typeinfo>
using namespace std;
class List { char elem; List * next;
             int ballast[30000000];
public:
List(const char*s): elem(*s? *s:'\n'),
          next(*s?  new List(s+1): NULL){}

List & operator*= (List & l) {
    List *q=&l, *p=this;
    while (p->next!=NULL) p=p->next;
    while (q->next!=NULL){
        p->elem = q->elem;
        p->next = new List("");
        q=q->next; p=p->next;
    }
    return *this;
}
```

```
 ~List(){ if(next!=NULL) {delete next;}  }
 };

 int main() {
     List w("xyz");
     try{
         try { throw '5'; throw 5;
         }
         catch (int) {cout<< "int ";}
         catch (char){cout<< "char ";
             w*=w; throw;
         }
      }
      catch (char) {cout<< "char"<< endl;}
      catch(exception & e){
          cout<<typeid(e).name() <<endl;
      }
      return 0;
}
```

Ответ:

10. Опишите шаблонную функцию *Reverse* (*a*), которая по контейнеру *a* (вектор, список, очередь и т.п.) строит в качестве значения новый контейнер, в котором элементы *a* расположены в обратном порядке. Например, для списка ⟨1, 2, 3⟩ функция построит список ⟨3, 2, 1⟩ .