Вариант_1_2025 Ф.И.О._____№ группы_____

NB! **Во всех задачах, где это требуется, предполагается наличие необходимых включений .**

1	2	3	4	5	6	7	8

1. Для каждого обращения к функции f() укажите, разрешим ли вызов, и что будет напечатано в случае положительного ответа.

```
struct A {
    operator int () { return 5; }
};
void f (int i, long int li) { std::cout << "::f()\n"; }
namespace NS1 { void f(bool b, long double ld) { std::cout << "NS1::f()\n"; } }
namespace NS2 { void f(char c, double d) { std::cout << "NS2::f()\n"; } }
using namespace NS2;
int main() {
    A a;
    f (1L, 1.0F);
    f (a, 3.5L);
    using namespace NS1;
    f (true, 1.5F);
}
```

2. В приведённой программе есть ошибки компиляции. ОБЪЯСНИТЕ в чем они заключаются и ИСПРАВЬТЕ код, приводящий к ошибке, **ничего не удаляя** каким-либо образом и НЕ МЕНЯЯ функцию main()! **Что будет напечатано** в результате работы получившейся программы?

```
struct B {
    int st;
    B () { std::cout << "  B::Constr\n";}
   ~B () { std::cout << "B::Destr\n"; }
    virtual void f() const { std::cout << "B::f()\n"; }
    virtual void g(int) const{ std::cout << "B::g(int)\n";}
};
class D : B {
public:
    D () { std::cout << "D::Constr\n"; }
   ~D () { std::cout << "D::Destr\n"; }
    virtual void f() { std::cout << "D::f()\n"; }
    void g(long int)const{ std::cout<< "D::g(long int)\n";}
};
int main() {
    const D d;
    const B *pb = &d;
    pb -> f();
    pb -> g(4);
    std::cout << ++D::st << std::endl;
}
```

3. Что будет напечатано в результате работы следующей программы?

```
using namespace  std;

struct A {
    A() {
        cout << "Constr ";
    }
    A(const A & lvr) {
        cout << "Copy_Constr ";
    }
   ~A() {
        cout << "Destr ";
    }
};
```

```
void f() {
    try { A obj;  throw obj;
    }
    catch(A&){ cout<<"f_Catch(A&) "; throw;
    }
}
int main() {
    try { A a; f(); throw a;
    }
    catch(A & ) { cout << "Catch(A&) ";
    }
    catch(...) { cout << "Catch(...) ";
    }
}
```

4. В приведённой программе есть ошибки компиляции. ОБЪЯСНИТЕ в чем они заключаются и ИСПРАВЬТЕ код, **ничего не удаляя** каким-либо образом, НЕ МЕНЯЯ функцию **main()** и структуры **A** и **B** и описывая **не более одной** новой функции.

```
template <class T>
int beg_end_cmp (T& cont) {
    T::iterator beg = cont.begin(), end = cont.end();
    return beg != end && *beg < *(--end) ? 1 : 2;
}
struct A {int a = 1;};
struct B {int a = 3;};
int main() {
    A a; B b;
    std::vector<A> va{a};
    std::vector<B> vb{b, b};
    std::cout << beg_end_cmp(va) + beg_end_cmp(vb) << std::endl;
}
```

5. Построить ПОЛИЗ для фрагмента программы на языке Си:

```
if (a>b) { a=b-1; } else { a=b=(b-1)*(a+1); }
```

Ответ:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ПОЛИЗ																	

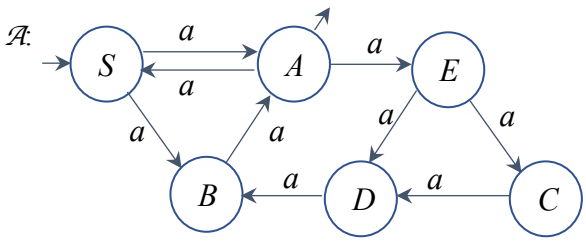
17	18	19	20	21	22	23	24	25	26

.

6. Построить регулярное выражение для языка L из всех цепочек в алфавите $\{a, b, c\}$, которые начинаются на ab и не оканчиваются на ab. Например, цепочки aba, $abba$, $abcba$ входят в L, а цепочки ε, a, ab не входят.
Длина записи регулярного выражения должна быть **не более** 30 символов. Знак конкатенации • в записи опускаем, например: для конкатенации a и b вместо $a \bullet b$ пишем просто ab.

Ответ:

7. По заданному конечному автомату $\mathcal{A}$ (на рисунке) построить эквивалентный ДКА $\mathcal{A}_{min}$ с минимальным числом состояний (вершин).
Пояснить, каким способом построено решение.



Ответ:

8. Рассмотрим язык троичных записей всех положительных целых чисел (без ведущих нулей):
L_3={1, 2, 10, 11, 12, 20, …}.

(**а**) Построить грамматику G_3, порождающую язык L_3, к которой примени́м метод рекурсивного спуска, или, иначе говоря – выполняется критерий LL(1)-грамматики. Применимость (выполнение критерия) обосновать.

(**б**) Используя глобальную булевскую переменную CF (*C*arry *F*lag – признак переноса в следующий разряд), добавить в грамматику G_3 действия по переводу (в процессе рекурсивного спуска) троичной записи числа n в обращение троичной записи числа n+1. Например, для входной цепочки 12 напечатается 02, так как она является обращением цепочки 20, и $20_3=12_3+1_3$. Других вспомогательных переменных и объектов (кроме CF) не использовать.

Ответ:

Ответы_Вариант 1_2025 (за каждую задачу максимум 10 баллов, минимум – 0)

1.
```
int main() {    A a;                                  // Перегрузки функций, пространства имен, using
                f(1L, 1.0F);    // NS2::f()
                f(a, 3.5L);     // ::f()
                using namespace NS1;
                f(true, 1.5F);   } // Вызов неразрешим
```

1.
```
        // Наследование, динамический полиморфизм, константные, статические члены класса
struct B {                                  // возможны варианты
        static int st;   // вместо static  возможно mutable и инициализация st в конструкторе
        B () { std:::cout << "  B::Constr\n";}
       ~B () { std::: cout << "B::Destr\n"; }
        virtual void f() const { std::: cout << "B::f()\n"; }
        virtual void g(int)const{ std::: cout << "B::g(int)\n";}
};
int B::st = 0;
class D : public B {
public:
         D () { std::cout << "  D::Constr\n"; }
        ~D () { std::cout << "D::Destr\n"; }
        virtual void f() { std::cout << "D::f()\n"; }
        void g(long int)const{ std:::cout<< "D::g(int)\n";} };
int main() {
        const  D d;             // B::Constr    D::Constr
        const B *pb = &d;       // Err! - недоступный базовый класс
        pb -> f();              // B::f()
        pb -> g(4);             // B::g(int)
        std::cout<<++D::st << std:::endl; //Err!– обращение к полю-данному через имя класса
                                // 1 (или другое инициализирующее значение +1) D::Destr   B::Destr
}
```

3. //Аппарат исключений, конструкторы в современном C++

Constr Constr Copy_Constr Destr f_Catch(A&) Destr Catch(A&) Destr

4.
```
template <class T>                                                  // Шаблоны, STL
bool operator< (const T& t1, const T& t2) {
        return t1.a  <  t2.a; // принимаем любую компилируемую реализацию <
}
template <class T>
int beg_end_cmp (T& cont) {
        typename T::iterator beg = cont.begin(), end = cont.end(); // пропущено typename
        return  beg != end && *beg < *(--end) ? 1 : 2;
}
struct A {int a = 1;};
struct B {int a = 3;};
int main() {
        A a;  B b;
        std::vector <A> va{a };
        std::vector<B> vb{b, b};
        std::cout << beg_end_cmp (va) + beg_end_cmp (vb) << std:: endl;
                                        // Err!-не определена операция < для A и B
}
```

5.

ПОЛИЗ

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	*b*	>	13	!F	&*a*	*b*	1	–	=	;	25	!	&*a*	&*b*	*b*	1

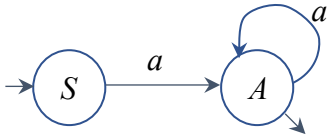
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
–	*a*	1	+	*	=	=	;									

Операция ';' выбрасывает из стека ставшее ненужным верхнее значение. Операция = в языке Си кроме присваивания значения переменной оставляет это значение на стеке как результат операции.

6. $ab(a+b+c)^*(a+c+bb+cb) + abb$. Вместо знака + можно использовать | .

7.

$\mathcal{A}_{мин}$:



Пояснение: можно заметить, что автомат $\mathcal{A}$ допускает все непустые цепочки в алфавите $\{a\}$. Построим «вручную» как можно более простой ДКА $\mathcal{A}_{мин}$. Если к нему применить какой-либо алгоритм минимизации ДКА (Бржозовского или Хопкрофта), то автомат не изменится (кроме названий состояний). Значит $\mathcal{A}_{мин}$ – это минимальный ДКА.

Возможны прямые применения к $\mathcal{A}$ алгоритма минимизации (Бржозовского) или сначала детерминизация с последующей минимизацией (Хопкрофта, или Бржозовского), – такие решения тоже засчитываем.

8. Рекурсивный спуск (LL(1)-анализатор) рассматривается для приведённых КС-грамматик (праволинейная приведённая грамматика тоже подходит), удовлетворяющих критерию.

G_3:	Обоснование применимости РС (выполнения LL(1)-критерия):
$S \to 1A \mid 2A$	$first(1A) \cap first(2A) = \varnothing$, нет вывода ε из альтернатив для S.
$A \to 0A \mid 1A \mid 2A \mid \varepsilon$	$first(1A) \cap first(2A) = \varnothing, first(2A) \cap first(3A) = \varnothing, first(1A) \cap first(3A) = \varnothing,$ $first(1A) \cap first(\varepsilon) = \varnothing,\ first(2A) \cap first(\varepsilon) = \varnothing,\ first(3A) \cap first(\varepsilon) = \varnothing,$ $first(A) \cap follow(A) = \varnothing.$ Возможны другие формы записи обоснования.

$G_{3\ translate}$:

$S \to 1A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 2 : 1\ );\ \rangle \mid\ 2A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, "01" : "2"\ );\ \rangle$

$A \to 0A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 1 : 0\ );\ \rangle \mid\ 1A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 2 : 1\ );\ \rangle \mid\ 2A \langle\ cout{<<}\ (CF\ ?\ 0 : 2\ );\ \rangle \mid$
$\varepsilon \langle\ CF=\textbf{\textit{true}};\ \rangle$

Действия можно выразить иначе (используя ***if*** , операторы-выражения и др.)

Возможны ответы с пополненной грамматикой (с новым начальным символом P и маркером конца \$):

G_3:	Установка $CF=\textbf{\textit{true}}$ может быть и в начале «спуска»:
$P \to S\$$	$P \to \langle\ CF=\textbf{\textit{true}};\ \rangle\ S\ \$$
$S \to 1A \mid 2A$	$S \to 1A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 2 : 1\ );\ \rangle \mid\ 2A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, "01": "2"\ );\ \rangle$
$A \to 0A \mid 1A \mid 2A \mid \varepsilon$	$A \to 0A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 1 : 0\ );\ \rangle \mid\ 1A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 2 : 1\ );\ \rangle \mid$ $2A \langle\ cout{<<}\ (CF\ ?\ 0 : 2\ );\ \rangle \mid\ \varepsilon$

Вариант_2_2025 Ф.И.О._____№ группы_____

NB! **Во всех задачах, где это требуется, предполагается наличие необходимых включений .**

1. Для каждого обращения к функции f() укажите, разрешим ли вызов, и что будет напечатано в случае положительного ответа.

1	2	3	4	5	6	7	8

```
struct A {
    A(int) { }
};
void f(int i, long int li) { std::cout << "::f()\n"; }
namespace NS1 { void f(bool b, long double ld) { std::cout << "NS1::f()\n";} }
namespace NS2 { void f(A a, const char * cstr) { std::cout << "NS2::f()\n";} }
using namespace NS2;
int main() {
    f(1L, 1.0F);
    f(3, "abc");
    using namespace NS1;
    f(true, 1.5f);
}
```

2. В приведённой программе есть ошибки компиляции. ОБЪЯСНИТЕ в чем они заключаются и ИСПРАВЬТЕ код, приводящий к ошибке, **ничего не удаляя** каким-либо образом и НЕ МЕНЯЯ функцию main()! **Что будет напечатано** в результате работы получившейся программы?

```
struct B {
    int st;
    B () { std::cout << "B::Constr\n";}
   ~B () { std::cout << "B::Destr\n"; }
    virtual void f() const { std::cout << "B::f()\n"; }
    virtual void g(int) { std::cout << "B::g(int)\n"; }

};
class D : public B {
public:
    D () { std::cout << "D::Constr\n"; }
   ~D () { std::cout << "D::Destr\n"; }
    virtual void f() { std::cout << "D::f()\n"; }
    void g(long int) const { std::cout<< "D::g(long int)\n"; }
};
int main() {
    const B *pb = new D;
    pb->f();
    pb->g(4);
    std::cout<<++(*pb).st << std::endl;
    delete pb;
}
```

3. Что будет напечатано в результате работы следующей программы?

```
using namespace  std;

struct A {
    A() {
        cout << "Constr ";
    }
    A(A && rvr) {
        cout << "Move_Constr ";
    }
    A(const A & lvr) = delete;
   ~A() {
        cout << "Destr ";
    }
};
```

```
void f() {
    try { A obj;  throw obj;
    }
    catch(...){ cout<<"f_Catch(…) "; throw;
    }
}
int main() {
    try { A a; f(); throw a;
    }
    catch (A & ) { cout << "Catch(A&) ";
    }
    catch (...) { cout << "Catch(…) ";
    }
}
```

4. В приведённой программе есть ошибки компиляции. ОБЪЯСНИТЕ в чем они заключаются и ИСПРАВЬТЕ код, **ничего не удаляя** каким-либо образом, НЕ МЕНЯЯ функцию **main()** и структуры **A** и **B** и описывая **не более одной** новой функции.

```
template <class T>
int edge_cmp(T& cont) {
    T::iterator ib = cont.begin(), ie = cont.end();
    return ib != ie && *ib > *(--ie) ? 0 : 1;
}

struct A {double d = 1.5;};
struct B {double d = 5.1;};
int main() {
    A a; B b;
    std::list<A> LA{a,a};
    std::list<B> LB{b,b,b};
    std::cout << edge_cmp(LA) - edge_cmp(LB) << std::endl;
}
```

5. Построить ПОЛИЗ для фрагмента программы на языке Си:

```
while (a>b) { a=a-1;  r=b=(b-1)*(a+1); }
```

Ответ:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ПОЛИЗ																	

17	18	19	20	21	22	23	24	25	26	
										∘

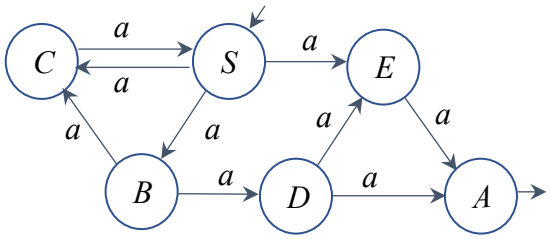
6. Построить регулярное выражение для языка L из всех цепочек в алфавите $\{a, b, c\}$, которые не начинаются на ab и оканчиваются на ab. Например, цепочки aab, $babab$, $aacab$ входят в L, а цепочки ε, a, ab не входят. Длина записи регулярного выражения должна быть **не более** 30 символов. Знак конкатенации • в записи опускаем, например: для конкатенации a и b вместо $a \bullet b$ пишем просто ab.

Ответ:

7. По заданному конечному автомату $\mathcal{A}$ (на рисунке) построить эквивалентный ДКА $\mathcal{A}_{min}$ с минимальным числом состояний (вершин).

Пояснить, каким способом построено решение.

$\mathcal{A}$:



Ответ:

8. Рассмотрим язык троичных записей всех положительных целых чисел (без ведущих нулей):

L_3={1, 2, 10, 11, 12, 20, …}.

(**а**) Построить грамматику G_3, порождающую язык L_3, к которой применим метод рекурсивного спуска, или, иначе говоря – выполняется критерий LL(1)-грамматики. Применимость (выполнение критерия) обосновать.

(**б**) Используя глобальную булевскую переменную *CF* (*C*arry *F*lag – признак заёма из следующего разряда), добавить в грамматику G_3 действия по переводу (в процессе рекурсивного спуска) троичной записи числа n в обращение троичной записи числа n-1. Например, для входной цепочки 21 напечатается 02, так как она является обращением цепочки 20, и $21_3 - 1_3 = 20_3$. Других вспомогательных переменных и объектов (кроме *CF*) не использовать. Еще пример: $10 \Rightarrow 20$, так как $10_3 - 1_3 = 02_3$, т.е. в результате «ведущий» ноль допускается.

Ответ:

Ответы_Вариант 2_2025 (за каждую задачу максимум 10 баллов, минимум - 0)

1. // Перегрузки функций, пространства имен, using

```
int  main() {      f(1L, 1.0F); // ::f()
                   f(3, "abc"); // NS2::f()
                   using namespace NS1;
                   f(true, 1.5f); } //NS1::f()
```

2. //Наследование, динамический полиморфизм, константные, статические члены класса

```
struct B {
        static int st;
        B () {std::cout << "  B::Constr\n";}
        virtual ~B () { std::cout << "B::Destr\n"; }
        virtual void f() const { std::cout << "B::f()\n"; }
        virtual void g(int) const{ std::cout << "B::g(int)\n";}
};
int B::st = 0;
class D : public B {
public:
         D () { std:: cout << "  D::Constr\n"; }
        ~D () { std::cout << "D::Destr\n"; }
         virtual void f () { std::cout << "D::f()\n"; }
         void g (long int)const{ std::cout<< "D::g(int)\n";} };
int main() {
        const B *pb = new D;    // B::Constr    D::Constr
         pb -> f();             // B::f()
         pb -> g(4); // Err! применение неконстантного метода к константному объекту //B::g(int)
         std::cout << ++(*pb).st << std::endl; // Err! – попытка изменить константный объект
                                  // 1 (или другое инициализирующее значение +1)
         delete pb; // или B::Destr, или делают деструктор виртуальным и D::Destr B::Destr
                    (если не сделали деструктор в B виртуальным, не снижаем)
}
```

3. //Аппарат исключений, конструкторы в современном C++

Constr Constr Move_Constr Destr f_Catch(...) Destr Catch(A&) Destr

4.

```
template <class T>                                                  // Шаблоны, STL
bool operator> (const T& t1, const T& t2) {
        return t1.d  >  t2.d; // принимаем любую компилируемую реализацию >
}
template <class T>
int edge_cmp (T& cont) {
        typename T::iterator ib = cont.begin(), ie = cont.end(); // Err!-пропущено typename
        return  ib != ie  &&  *ib > *(--ie)  ?  0  :  1;
}
struct A {double d = 1.5;};
struct B {double d = 5.1;};
int main() {
        A a;  B b;
        std::vector <A> va{a };
        std::vector<B> vb{b, b};
        std::cout << edge_cmp (va) - edge_cmp (vb) << std::endl; // Err!-нет операции  > для A и B
}
```

5.

ПОЛИЗ	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	a	*b*	>	25	!F	&*a*	*a*	*1*	–	=	;	&*r*	&*b*	*b*	1	–	*a*

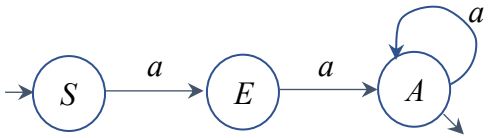
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	+	*	=	=	;	0	!									

Операция ';' выбрасывает из стека ставшее ненужным верхнее значение. Операция = в языке Си кроме присваивания значения переменной оставляет это значение на стеке как результат операции.

6. $(b+c+aa+ac)(a+b+c)^*ab + aab$. Вместо знака + можно использовать | .

7.

$\mathcal{A}_{мин}$:



Пояснение: можно заметить, что автомат $\mathcal{A}$ допускает все цепочки в алфавите $\{a\}$ с длиной больше единицы. Построим «вручную» как можно более простой ДКА $\mathcal{A}_{мин}$. Если к нему применить какой-либо алгоритм минимизации ДКА (Бржозовского или Хопкрофта), то автомат не изменится (кроме названий состояний). Значит $\mathcal{A}_{мин}$ – это минимальный ДКА.

Возможны прямые применения к $\mathcal{A}$ алгоритма минимизации (Бржозовского) или сначала детерминизация с последующей минимизацией (Хопкрофта, или Бржозовского), – такие решения тоже засчитываем.

8. Рекурсивный спуск (LL(1)-анализатор) рассматривается для приведённых КС-грамматик (праволинейная приведённая грамматика тоже подходит), удовлетворяющих критерию.

G_3:
$S \to 1A \mid 2A$
$A \to 0A \mid 1A \mid 2A \mid \varepsilon$

Обоснование применимости РС (выполнения LL(1)-критерия):
$first(1A) \cap first(2A) = \varnothing$, нет вывода ε из альтернатив для S.
$first(1A) \cap first(2A) = \varnothing, first(2A) \cap first(3A) = \varnothing, first(1A) \cap first(3A) = \varnothing,$
$first(1A) \cap first(\varepsilon) = \varnothing,\ first(2A) \cap first(\varepsilon) = \varnothing,\ first(3A) \cap first(\varepsilon) = \varnothing,$
$first(A) \cap follow(A) = \varnothing.$
Возможны другие формы записи обоснования.

$G_{3\ translate}$:
$S \to 1A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 0 : 1\ );\ \rangle \mid 2A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 1 : 2\ );\ \rangle$
$A \to 0A \langle\ cout{<<}\ (CF\ ?\ 2 : 0\ );\ \rangle \mid 1A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}},\ 0\ : 1\ );\ \rangle \mid 2A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 1 : 2\ );\ \rangle \mid$
$\varepsilon \langle\ CF=\textbf{\textit{true}};\ \rangle$
Действия можно выразить иначе (используя ***if***, операторы-выражения и др.)

Возможны ответы с пополненной грамматикой (с новым начальным символом P и маркером конца \$):

G_3:
$P \to S\$$
$S \to 1A \mid 2A$
$A \to 0A \mid 1A \mid 2A \mid \varepsilon$

Установка $CF=\textbf{\textit{true}}$ может быть и в начале «спуска»:
$P \to \langle\ CF=\textbf{\textit{true}};\ \rangle\ S\ \$$
$S \to 1A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 0 : 1\ );\ \rangle \mid 2A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 1 : 2\ );\ \rangle$
$A \to 0A \langle\ cout{<<}\ (CF\ ?\ 2 : 0\ );\ \rangle \mid 1A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 0 : 1\ );\ \rangle \mid$
$2A \langle\ cout{<<}\ (CF\ ?\ CF=\textbf{\textit{false}}, 1 : 2\ );\ \rangle \mid \varepsilon$