Экзамен_2021_2 ФИО_____№ группы_____

- ✓ Проанализировать следующие грамматики:

1. $S \to 0A \mid 1S$ $A \to 1A \mid 1$
2. $S \to aSBc \mid \varepsilon$ $B \to bB \mid c$
3. $S \to A0 \mid S1$ $A \to A1 \mid 0$
4. $S \to cASa \mid \varepsilon$ $A \to aA \mid \varepsilon$

Никак **не преобразовывая** грамматики, выбрать одну из них для построения по грамматике корректного анализатора **методом рекурсивного спуска**, а другую - для анализатора по **конечному автомату**.
Выбор **ОБОСНОВАТЬ**. *(По 10 баллов за каждый правильный и правильно обоснованный ответ)*

- ✓ Какие **языки** порождают выбранные грамматики? *(По 5 баллов за каждый правильный ответ)*
- ✓ Каков **тип** этих языков по Хомскому? *(По 5 баллов за каждый правильный ответ)*

- ✓ Описать **два** класса ***LexAn*** и ***SyntAn***, каждый из которых - **пропзводный** от заданного класса ***Base***: *(По 25 баллов за каждый правильно в соответствие условию задачи описанный класс)*

```
struct Base {
   static int n;
   Base (int t) { n *= t;}
   virtual void gc ( ) = 0;                  // получает очередной символ анализируемой цепочки
   virtual const char * operator * (const char * str) = 0;     // выполняет анализ цепочки str
                    // по выбранной грамматике, возвращает строку-сообщение о
                    // принадлежности (или нет) цепочки языку, порождаемому грамматикой.
    virtual ~ Base ( ) { cout << n << endl;}
};
int Base::n = 1;
```

- ✓ а также **одну** функцию *analyse (**const char** * str), запускающую анализатор по входной цепочке, (10 баллов за правильно описанную функцию)*
такие, что в следующей функции *main ( )* не будет ошибок:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string  str;
    cin >> str;                          // ввод цепочки для анализа по конечному автомату
    const char * s = str.c_str();        // возвращает указатель на строку(цепочку) в смысле языка Си
    analyse <LexAn> (s);

    cin >> str;                          // ввод цепочки для анализа методом рекурсивного спуска
    s = str.c_str();
    analyse <SyntAn> (s);
    return 0;
}
```

а при выполнении программы на экран будет выдано:

```
Yes!!!          // или   No! – в случае непринадлежности цепочки языку
Lucky number – 7
Success!!!      // или   Bad string! – в случае непринадлежности цепочки языку
Lucky number – 777
```

Экзамен_2021_2. Ответы и критерии оценки
Исходный балл - **100**

1. Выбираем для анализа по КА регулярную грамматику, для которой автомат детерминирован – это №3 (другие или не регулярные, или анализ по ним недетерминированный)
   - неверный выбор (нет обоснования) - **-10**
2. Для анализа РС-методом выбираем КС-грамматику, к которой метод применим, это № 2 (к другим метод неприменим: левая рекурсия, плохая ε-альтернатива, пересечение множеств *first(...)* в альтернативах)
   - неверный выбор (нет обоснования) - **-10**

1) $\begin{cases} S \to 0A \mid 1S \\ A \to 1A \mid 1 \end{cases}$ 2) $\begin{cases} S \to aSBc \mid \varepsilon \\ B \to bB \mid c \end{cases}$ 3) $\begin{cases} S \to A0 \mid S1 \\ A \to A1 \mid 0 \end{cases}$ 4) $\begin{cases} S \to cASa \mid \varepsilon \\ A \to aA \mid \varepsilon \end{cases}$

3. $L(2) = \{ a^n (b^{m_n} cc)^n \mid n >= 0, m_n >= 0\}$ - **-5**, тип 2 - **-5**
4. $L(3) = \{ 0\ 1^n\ 0\ 1^m \mid n, m >= 0\}$ - **-5**, тип 3 - **-5**

```
class LexAn : public Base {
        const char * p;
        char c;
        void gc() { c = *p++; }                          // нет функции - -10, неверная - -5
public:
        LexAn () : Base (7){ }          // нет конструктора или списка инициализации -  -10
        const char * operator * (const char * str) {          // неверный алгоритм -  -15,
           try{ p = str;                                      // мелкие ошибки -  -2 за каждую
                enum state {H, A, S} CS;
                CS = H;
                do {    gc();
                        switch(CS){
                                case H: if (c == '0') CS = A;
                                        else  throw c;
                                        break;
                                case A: if (c == '0') CS = S;
                                        else
                                                if (c == '1');
                                                else throw c;
                                        break;
                                case S: if (c == '1');
                                        else
                                                if (c == '\0')
                                                        return "Yes!";
                                                else   throw c;
                        }
                }
                while(true);
           }
           catch(char){return "NO!";}
        }
        ~LexAn() { cout << "Magic number - "; }

};
```

```
class SyntAn : public Base {
        const char * p;
        void gc() { c = *p++; }                    // нет функции - -10, неверная - -5
        char c;
        void S() {
                if(c =='a'){
                        gc(); S(); B() ;
                        if(c =='c')
                                gc();
                        else
                                throw c;
                }
        }
        void B() {
                if(c =='b'){  gc();  B();
                }
                else
                        if (c == 'c') gc();
                        else
                                throw c;
        }                                          // ошибки в РС-методе - -5 за каждую
public:
        SyntAn() : Base (111) { }                  // ошибки как в LexAn – караются 1 раз
        ~SyntAn() {cout << "Lucky number - ";}
        const char * operator * (const char * str) {        // неверный алгоритм - -15,
          try { p = str;                                    // мелкие ошибки - -2 за каждую
                gc();
                S();
                if (c == '\0')
                        return "Succsess!!!";
                else
                        throw c;
          }
          catch(char) { return "Bad string!"; }
        }
};

template <class T>                         // неверная функция - -10
void  analyse (const char * str) {
        T  t;
        cout << t * str  << endl;
}
```

Неверный вывод - **-5** за каждый
(1 и/или 3, 2 и/или 4 неверные строки вывода караются по одному разу)