Фамилия Имя_____Группа_____Вар1

1	2	3	4	5	6	7	8

*Во всех задачах на C++ предполагается наличие необходимых директив #include <...> и оператора **using namespace** std;*

1. (15 баллов) Что будет выдано на печать в результате работы следующей программы?

```
struct A {
        A(){ cout << 1 ;}
        A(const A&){ cout << 2;}
        virtual ~A(){ cout << 3;}
};
```

```
struct B : A {
        B(){ cout << 4;}
        B(const B&){ cout << 5;}
        ~B(){ cout << 6;}
};
```

```
void f(A & a, B & b){
        try {   if (typeid(a) == typeid(b))  throw a;
                else   throw b;
        }
        catch (B&) { cout << "catch(B&)\n";  b = dynamic_cast <B&> (a); }
        catch (A&) { cout << "catch(A&)\n";  b = dynamic_cast <B&> (a); }
}
int main() {    try {    A a;  B b;
                         f (a, b);
                 }
                 catch (...) {cout << "catch(...)";}
                 cout << "End\n";  return 0;
}
```

2. (20 баллов) Опишите необходимые **класс *A*** и булевскую функцию ***fun*** от двух параметров (контейнера-списка и целого), определяющую, есть ли в списке элемент, соответствующий заданному целому, так, чтобы в приведенной ниже функции *main ()* не было ошибок.

```
int main() {
        const list<A> L {/* инициализация списка*/};
        cout << A::n << fun<A> (L, 2) << endl;
}
```

3. (10 баллов) Модифицируйте **только** описание классов A и B, **ничего в них не удаляя каким-либо образом**, так, чтобы в приведенной ниже функции *main()* не было ошибок и ситуаций *undefined behavior* и напечаталось: ConstrA ConstrB B::f() DestrB DestrA

```
struct A {
        A(int a){ cout << "ConstrA ";}
        ~A() { cout << "DestrA "; }
        virtual void g() = 0;
        void f(){ cout << "A::f() ";}
};
struct B : A {
        B(){ cout << "ConstrB ";}
        ~B() { cout << "DestrB "; }
        void f(){ cout << "B::f() ";}
};
int main() {
        A & p = *new B();
        p.f();
        delete  &p;
}
```

4. (5 баллов) Приведенная ниже программа не компилируется. Укажите **причины** ошибок и **модифицируйте <u>двумя способами</u> только** описание класса *A* так, чтобы ошибок не было. **Что будет напечатано** в результате работы получившейся программы?

```
class A {  int x, y;
public:
        A(int a = 5){ x = y = a; }
};
namespace N1 {
        void f (int) { cout << "N1::f(int)\n"; }
}
void f(const A & a) { cout << a.x << a.y << "f(A)\n"; }
int main() {      A a;  f(a);  f(2); }
```

5. (10 баллов) Существует ли приведённая КС-грамматика с единственным правилом вывода (без альтернатив), к которой метод рекурсивного спуска неприменим? Обоснуйте ответ.

6. (5 баллов) Дайте определение понятию *система программирования*.

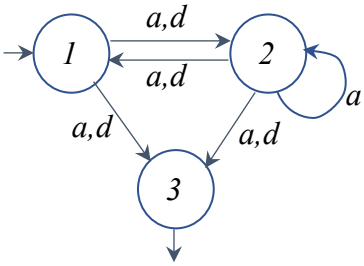
7. (20 баллов) В префиксной записи выражения сначала идет знак операции, затем операнды. Например, префиксная запись выражения *a*(a+a)* такова: **a+aa*. Дана заготовка грамматики с действиями C++. Добавьте в нее недостающие действия так, чтобы анализатор, построенный методом рекурсивного спуска, переводил инфиксную (обычную) запись в префиксную, например: *a*a*(a+a)* => ***aa+aa*. Анализатору доступен глобальный стек `stack<string> st;` и своя локальная переменная `string s;` в каждой рекурсивной функции *S, A, M, F*. В качестве добавляемых действий допускаются только операторы-выражения, содержащие операции со стеком и/или арифметические операции с присваиванием.

$S \to \langle$ `while(!st.empty()) st.pop();` $\rangle A \langle$ `s = st.top(); st.pop(); cout << s;` $\rangle$
$A \to M \ \{ + M \}$
$M \to F \ \{ * F \}$
$F \to (A) \mid a \ \langle$ `s = "a"; st.push(s);` $\rangle$

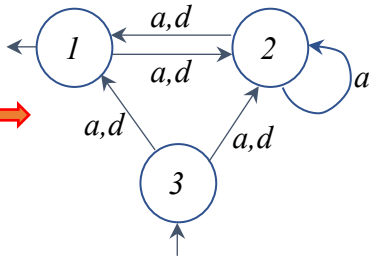
8. (15 баллов) Обращением $rev(\mathcal{A})$ конечного автомата $\mathcal{A}$ будет конечный автомат, у которого все начальные (входные) вершины $\mathcal{A}$ становятся заключительными (выходными), в выходные – входными, и направления дуг заменяются на противоположные (см. рисунок). Через $det(\mathcal{A})$ обозначим результат детерминизации автомата $\mathcal{A}$.

По заданному автомату $\mathcal{A}$ постройте эквивалентный и минимальный по числу состояний ДКА $\mathcal{A}_{min}$, используя алгоритм Бржозовского: $\mathcal{A}_{min} := det\,(rev\,(det\,(rev\,(\mathcal{A})\,)\,)\,)$.

$\mathcal{A}$:



$rev(\mathcal{A})$:



$det(rev(\mathcal{A}))$:

$rev(det(rev(\mathcal{A})))$:

$det(rev(det(rev(\mathcal{A}))))$:

$\mathcal{A}_{min}$:

1	2	3	4	5	6	7	8

Фамилия Имя_____Группа_____Вар2

Во всех задачах на C++ предполагается наличие необходимых директив #include <...> и оператора ***using namespace*** *std;*

1. (15 баллов) Что будет выдано на печать в результате работы следующей программы?

```
struct A {
        A(){ cout << 1 ;}
        A(const A&){ cout << 2;}
        virtual ~A(){ cout << 3;}
};
```

```
struct B : A {
        B(){ cout << 4;}
        B(const B&){ cout << 5;}
        ~B(){ cout << 6;}
};
```

```
void f (A & a, B & b){
        try {   if (typeid(a) == typeid(b))  throw a;
                else   throw b;
        }
        catch (B&){ cout << "catch(B&)\n";  b = dynamic_cast <B&> (a); }
        catch (A&){ cout << "catch(A&)\n";  b = dynamic_cast <B&> (a); }
}

int main() {    try {    A a;   B b;   f (b, b);
                }
                catch (...) {cout << "catch(...)";}
                cout << "End\n";   return 0;
}
```

2. (20 баллов) Опишите необходимые класс *A* и функцию *count* от двух параметров (контейнера-списка и целого), подсчитывающую количество элементов списка, соответствующих заданному целому, так, чтобы в приведенной ниже функции *main ()* не было ошибок.

```
        int main() {
                const list<A> L {/* инициализация списка*/};
                cout << A::n << count <A> (L, 2) << endl;
        }
```

3. (10 баллов) Модифицируйте **только** описание классов A и B, **ничего в них не удаляя каким-либо образом**, так, чтобы в приведенной ниже функции *main()* не было ошибок и ситуаций *undefined behavior* и напечаталось : **ConstrA ConstrB B::g() DestrB DestrA**

```
struct A {
        A(int a){ cout << "ConstrA ";}
        ~A() { cout << "DestrA "; }
        virtual void f() = 0;
        void g(){ cout << "A::g() ";}
};
struct B : A {
        B(){ cout << "ConstrB ";}
        ~B() { cout << "DestrB "; }
        void g(){ cout << "B::g() ";}
};
int main() {
        A * p = new B();
        p -> f();
        delete  p;
}
```

4. (5 баллов) Приведенная ниже программа не компилируется. Укажите **причины** ошибок и **модифицируйте <u>двумя способами</u> только** описание класса *A* так, чтобы ошибок не было.
Что будет напечатано в результате работы получившейся программы?

```
class A {   int x,y;
public:
        A(int a = 5){ x = y = a; }
};
namespace N1 {
        void f (int) { cout << "N1::f(int)\n"; }
}
void f(const A & a) { cout << a.x << a.y << "f(A)\n"; }
using namespace N1;
int main() {      A a;  f(a);  f(2); }
```

5. (10 баллов) Существует ли приведённая КС-грамматика, которая порождает единственную цепочку, имеет два правила вывода (считая альтернативы), и к которой метод рекурсивного спуска неприменим? Обоснуйте ответ.

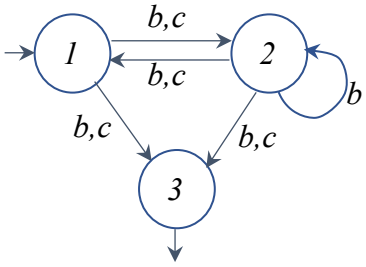
6. (5 баллов) Дайте определение понятию *программный продукт*.

7. (20 баллов) В префиксной записи выражения сначала идет знак операции, затем операнды. Например, префиксная запись арифметического выражения *a*(a+a)* такова: **a+aa*. Постфиксная запись (ПОЛИЗ) выражения *a*(a+a)* выглядит как *aaa+**, а обращение постфиксной записи *aaa+** таково: **+aaa*. Дана заготовка грамматики с действиями C++. Грамматика порождает обращения постфиксных записей. Добавьте в нее недостающие действия так, чтобы анализатор, построенный методом рекурсивного спуска, переводил обращение постфиксной записи в префиксную, например: ***+aaaa => *a*a+aa*. Анализатору доступен глобальный стек `stack<string> st;` и своя локальная переменная `string s;` в каждой рекурсивной функции *S* и *A*. В качестве добавляемых действий допускаются только операторы-выражения, содержащие операции со стеком и/или арифметические операции с присваиванием.

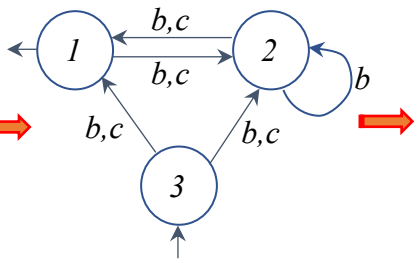
$S \to$ ⟨ `while(!st.empty()) st.pop();` ⟩ A ⟨ `s = st.top(); st.pop(); cout << s;` ⟩
$A \to + A\ A \mid * A\ A \mid a$ ⟨ `s = "a"; st.push(s);` ⟩

8. (15 баллов) Обращением *rev*($\mathcal{A}$) конечного автомата $\mathcal{A}$ будет конечный автомат, у которого все начальные (входные) вершины $\mathcal{A}$ становятся заключительными (выходными), в выходные – входными, и направления дуг заменяются на противоположные (см. рисунок). Через *det*($\mathcal{A}$) обозначим результат детерминизации автомата $\mathcal{A}$.
По заданному автомату $\mathcal{A}$ постройте эквивалентный и минимальный по числу состояний ДКА $\mathcal{A}_{min}$, используя алгоритм Бржозовского: $\mathcal{A}_{min} := det\,(rev\,(det\,(rev\,(\mathcal{A})\,)\,)\,)$.

$\mathcal{A}$:



rev($\mathcal{A}$):



det(*rev*($\mathcal{A}$)):

rev(*det*(*rev*($\mathcal{A}$))):

det(*rev*(*det*(*rev*($\mathcal{A}$)))):

$\mathcal{A}_{min}$: