6.1

Дадена е граматика за дефиниране на аритметични изрази:

p1: израз => израз + терм | израз – терм | терм
p2: терм => терм * фактор | терм / фактор | фактор
p3: фактор => (израз) | число
p4: число => цифра$^{+}$
p5: цифра => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следните низове:

2 + 2

2 + 2 * 2

(2 + 2)*2

3. Да се разработи програма на C++ за анализ и изчисляване на аритметични изрази.

Примерен вход:	Изход:
2 + 3 * 4	14
(2 + 3) * 4	20
(2 + 2*3	Грешка

Приоритет на операциите:

1. Скоби
2. Умножение, деление
3. Събиране, изваждане

6.2

Дадена е граматика за дефиниране на аритметични изрази:

p1: израз => израз + терм | израз – терм | терм
p2: терм => терм * фактор | терм / фактор | фактор
p3: фактор => (израз) | число
p4: число => цифра$^{+}$
p5: цифра => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следните низове:

2 + 2

2 + 2 * 2

(2 + 2)*2

3. Да се разработи програма на Java за анализ и изчисляване на аритметични изрази.

Примерен вход:	Изход:
2 + 3 * 4	14
(2 + 3) * 4	20
(2 + 2*3	Грешка

Приоритет на операциите:

1. Скоби
2. Умножение, деление
3. Събиране, изваждане

6.3

Дадена е граматика за дефиниране на аритметични изрази:

p1: израз => израз + терм | израз – терм | терм
p2: терм => терм * фактор | терм / фактор | фактор
p3: фактор => (израз) | число
p4: число => цифра$^{+}$
p5: цифра => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Задачи:

Да се използва YACC (или BISON) за генериране на анализатор за аритметични изрази.

Примерен вход:	Изход:
2 + 3 * 4	14
(2 + 3) * 4	20

6.4

Дадена е граматика за дефиниране на аритметични изрази:

p1: израз => израз + терм | израз – терм | терм
p2: терм => терм * фактор | терм / фактор | фактор
p3: фактор => (израз) | число
p4: число => цифра$^{+}$
p5: цифра => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Задачи:

Да се използва ANTLR (Another Tool for Language Recognition) за генериране на анализатор за аритметични изрази.

Примерен вход:	Изход:
2 + 3 * 4	14
(2 + 3) * 4	20

6.5.

Дадена е граматика за дефиниране на индексирана променлива:

p1: <индексирана променлива> => <променлива>[<израз>]
p2: <променлива> => id1 | id2
p3: <израз> => <прост израз>
p4: <прост израз> => <терм> (<оператор за събиране> <терм>)*
p5: <терм> => <фактор> (<оператор за умножение> <фактор>)*
p6: <фактор> => променлива | константа | (<израз>)
p7: <константа> => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
p8: <оператор за събиране> => + | -
p9: <оператор за умножение> => *

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следните низове:

id1[5]

id1[(2+2)*id2]

3. Да се разработи програма на C++, която получава входен текст и извежда резултат дали този текст е валидна индексирана променлива според правилата на дадената граматика.

Примерен вход:	Изход:
id1[5]	correct
id1[(2+2)*2]	correct
id1[]	not correct
id1[(2+]	not correct

6.6

Дадена е граматика за дефиниране на индексирана променлива:

p1: <индексирана променлива> => <променлива>[<израз>]
p2: <променлива> => id1 | id2
p3: <израз> => <прост израз>
p4: <прост израз> => <терм> (<оператор за събиране> <терм>)*
p5: <терм> => <фактор> (<оператор за умножение> <фактор>)*
p6: <фактор> => променлива | константа | (<израз>)
p7: <константа> => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
p8: <оператор за събиране> => + | -
p9: <оператор за умножение> => *

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следните низове:

id1[5]

id1[(2+2)*id2]

3. Да се разработи програма на Java, която получава входен текст и извежда резултат дали този текст е валидна индексирана променлива според правилата на дадената граматика.

Примерен вход:	Изход:
id1[5]	correct
id1[(2+2)*2]	correct
id1[]	not correct
id1[(2+]	not correct

6.7

Дадена е граматика за дефиниране на оператор IF:

p1: <оператор IF> => IF <израз> THEN <оператор>
| IF <израз> THEN <оператор> ELSE <оператор>
p2: <оператор> => statement
p3: <израз> => <прост израз> | <прост израз><условен оператор><прост израз>
p4: <прост израз> => <терм> (<оператор за събиране> <терм>)*
p5: <терм> => <фактор> (<оператор за умножение> <фактор>)*
p6: <фактор> => променлива | константа | (<израз>)
p7: <константа> => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
p8: <променлива> => id1 | id2
p9: <условен оператор> => = | <> | < | <= | >= | >
p10: <оператор за събиране> => OR
p11: <оператор за умножение> => AND

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следното изречение:

IF (id1>0 AND id1<10) OR (id2>0 AND id2<10) THEN statement ELSE statement

3. Да се разработи програма на C++, която получава входен текст и извежда резултат дали този текст е валиден оператор IF според правилата на дадената граматика.

Примерен вход:	Изход:
IF (id1>id2) THEN statement	correct
IF (id1>0 AND id1<10) THEN statement ELSE statement	correct
IF (id1>0 AND id1<10) OR (id2>0 AND id2<10) THEN statement ELSE statement	correct
IF (id1>0 AND id1<10) OR (id2>0 AND id2<10) THEN statement ELSE	not correct
IF (id1>0 AND id1<10) OR THEN statement ELSE statement	not correct

6.8

Дадена е граматика за дефиниране на оператор IF:

p1: <оператор IF> => IF <израз> THEN <оператор>
| IF <израз> THEN <оператор> ELSE <оператор>
p2: <оператор> => statement
p3: <израз> => <прост израз> | <прост израз><условен оператор><прост израз>
p4: <прост израз> => <терм> (<оператор за събиране> <терм>)*
p5: <терм> => <фактор> (<оператор за умножение> <фактор>)*
p6: <фактор> => променлива | константа | (<израз>)
p7: <константа> => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
p8: <променлива> => id1 | id2
p9: <условен оператор> => = | <> | < | <= | >= | >
p10: <оператор за събиране> => OR
p11: <оператор за умножение> => AND

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следното изречение:

IF (id1>0 AND id1<10) OR (id2>0 AND id2<10) THEN statement ELSE statement

3. Да се разработи програма на Java, която получава входен текст и извежда резултат дали този текст е валиден оператор IF според правилата на дадената граматика.

Примерен вход:	Изход:
IF (id1>id2) THEN statement	correct
IF (id1>0 AND id1<10) THEN statement ELSE statement	correct
IF (id1>0 AND id1<10) OR (id2>0 AND id2<10) THEN statement ELSE statement	correct
IF (id1>0 AND id1<10) OR (id2>0 AND id2<10) THEN statement ELSE	not correct
IF (id1>0 AND id1<10) OR THEN statement ELSE statement	not correct

6.9

Дадена е граматика за дефиниране на оператор WHILE:

p1: <оператор WHILE> => WHILE <израз> DO <оператор>
p2: <оператор> => statement
p3: <израз> => <прост израз> | <прост израз><условен оператор><прост израз>
p4: <прост израз> => <терм> (<оператор за събиране> <терм>)*
p5: <терм> => <фактор> (<оператор за умножение> <фактор>)*
p6: <фактор> => променлива | константа | (<израз>)
p7: <константа> => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
p8: <променлива> => id1 | id2
p9: <условен оператор> => = | <> | < | <= | >= | >
p10: <оператор за събиране> => OR
p11: <оператор за умножение> => AND

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следното изречение:

WHILE id1>0 AND id1>id2 DO statement

3. Да се разработи програма на C++, която получава входен текст и извежда резултат дали този текст е валиден оператор WHILE според правилата на дадената граматика.

Примерен вход:	Изход:
WHILE id1>id2 DO statement	correct
WHILE (id1>0 AND id1<10) DO statement	correct
WHILE (id1>0 AND id1<10) OR (id2>0 AND id2<10) DO statement	correct
WHILE (id1>0 AND id1<10) DO	not correct
WHILE (id1>0 AND id1<10) OR DO statement	not correct

6.10

Дадена е граматика за дефиниране на оператор WHILE:

p1: <оператор WHILE> => WHILE <израз> DO <оператор>
p2: <оператор> => statement
p3: <израз> => <прост израз> | <прост израз><условен оператор><прост израз>
p4: <прост израз> => <терм> (<оператор за събиране> <терм>)*
p5: <терм> => <фактор> (<оператор за умножение> <фактор>)*
p6: <фактор> => променлива | константа | (<израз>)
p7: <константа> => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
p8: <променлива> => id1 | id2
p9: <условен оператор> => = | <> | < | <= | >= | >
p10: <оператор за събиране> => OR
p11: <оператор за умножение> => AND

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следното изречение:

WHILE id1>0 AND id1>id2 DO statement

3. Да се разработи програма на Java, която получава входен текст и извежда резултат дали този текст е валиден оператор WHILE според правилата на дадената граматика.

Примерен вход:	Изход:
WHILE id1>id2 DO statement	correct
WHILE (id1>0 AND id1<10) DO statement	correct
WHILE (id1>0 AND id1<10) OR (id2>0 AND id2<10) DO statement	correct
WHILE (id1>0 AND id1<10) DO	not correct
WHILE (id1>0 AND id1<10) OR DO statement	not correct

6.11

Дадена е граматика за дефиниране на оператор WRITE:

p1: <оператор WRITE> => WRITE (<изходна стойност> (, <изходна стойност>)*)
p2: <изходна стойност> => <израз>
p3: <израз> => <прост израз> | <прост израз><условен оператор><прост израз>
p4: <прост израз> => <терм> (<оператор за събиране> <терм>)*
p5: <терм> => <фактор> (<оператор за умножение> <фактор>)*
p6: <фактор> => променлива | константа | (<израз>)
p7: <константа> => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
p8: <променлива> => id1 | id2
p9: <условен оператор> => = | <> | < | <= | >= | >
p10: <оператор за събиране> => OR
p11: <оператор за умножение> => AND

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следните изречения:

WRITE(id1+id2)

WRITE(id1>0 AND id1<10 OR id1>id2)

3. Да се разработи програма на C++, която получава входен текст и извежда резултат дали този текст е валиден оператор WRITE според правилата на дадената граматика.

Примерен вход:	Изход:
WRITE(id1)	correct
WRITE(id1+id2)	correct
WRITE (id1>0 AND id1<10)	correct
WRITE (id1	not correct
WRITE (id1>0 AND id1<10 OR)	not correct

6.12

Дадена е граматика за дефиниране на оператор WRITE:

p1: <оператор WRITE> => WRITE (<изходна стойност> (, <изходна стойност>)*)
p2: <изходна стойност> => <израз>
p3: <израз> => <прост израз> | <прост израз><условен оператор><прост израз>
p4: <прост израз> => <терм> (<оператор за събиране> <терм>)*
p5: <терм> => <фактор> (<оператор за умножение> <фактор>)*
p6: <фактор> => променлива | константа | (<израз>)
p7: <константа> => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
p8: <променлива> => id1 | id2
p9: <условен оператор> => = | <> | < | <= | >= | >
p10: <оператор за събиране> => OR
p11: <оператор за умножение> => AND

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следните изречения:

WRITE(id1+id2)

WRITE(id1>0 AND id1<10 OR id1>id2)

3. Да се разработи програма на Java, която получава входен текст и извежда резултат дали този текст е валиден оператор WRITE според правилата на дадената граматика.

Примерен вход:	Изход:
WRITE(id1)	correct
WRITE(id1+id2)	correct
WRITE (id1>0 AND id1<10)	correct
WRITE (id1	not correct
WRITE (id1>0 AND id1<10 OR)	not correct

5.1.

Да се разработи програма анализатор на език C++, която разделя думите в програма на C и разпознава следните типове за всяка дума:

- ключова дума – напр. if, for, while
- идентификатор – имена на променливи, имена на типове данни, имена на функции
- константи – напр. 0, 1, 2 и т.н.
- оператори – напр. +,-,*,/,=,>,>=,<=,==
- разделители – (,),;

Резултатът от изпълнението на анализатора ще бъде двойки стойности <дума, тип>.

Примерен вход:

```
int func_change()
{
    if (a>b)
            temp = a;
            a = b;
            b = temp;
    return 0;
}
```

Изход:
<int, идентификатор>
<func_change, идентификатор>
<(, разделител>
<), разделител>
<{, разделител>
<if, ключова дума>
<(, разделител>
<a, идентификатор>
<‘>‘, оператор>
и т.н.

Анализаторът ще работи само с целочислени и символни константи.

5.2.

Да се разработи програма анализатор на език Java, която разделя думите в програма на C и разпознава следните типове за всяка дума:

- ключова дума – напр. if, for, while
- идентификатор – имена на променливи, имена на типове данни, имена на функции
- константи – напр. 0, 1, 2 и т.н.
- оператори – напр. +,-,*,/,=,>,>=,<=,==
- разделители – (,),;

Резултатът от изпълнението на анализатора ще бъде двойки стойности <дума, тип>.

Примерен вход:

```
int func_change()
{
    if (a>b)
            temp = a;
            a = b;
            b = temp;
    return 0;
}
```

Изход:
<int, идентификатор>
<func_change, идентификатор>
<(, разделител>
<), разделител>
<{, разделител>
<if, ключова дума>
<(, разделител>
<a, идентификатор>
<‘>‘, оператор>
и т.н.

Анализаторът ще работи само с целочислени и символни константи.

5.3.

Да се разработи програма анализатор на език C++, която разделя думите в програма на Java и разпознава следните типове за всяка дума:

- ключова дума – напр. if, for, while
- идентификатор – имена на променливи, имена на типове данни, имена на функции
- константи – напр. 0, 1, 2 и т.н.
- оператори – напр. +,-,*,/,=,>,>=,<=,==
- разделители – (,),;

Резултатът от изпълнението на анализатора ще бъде двойки стойности <дума, тип>.

Примерен вход:

```
public class HelloWorld {
  public static void main() {
    System.out.println(10);
  }
}
```

Изход:
<public, ключова дума>
<class, ключова дума >
<HelloWorld, идентификатор>
<{, разделител>
<public, ключова дума>
<static, ключова дума>
…
<10, константа>
и т.н.

Анализаторът ще работи само с целочислени и символни константи.
Изход:
<int, идентификатор>
<func_change, идентификатор>
<(, разделител>
<), разделител>
<{, разделител>
<if, ключова дума>
<(, разделител>
<a, идентификатор>
<‘>‘, оператор>
и т.н.

Анализаторът ще работи само с целочислени и символни константи.

5.4.

Да се разработи програма анализатор на език Java, която разделя думите в програма на Java и разпознава следните типове за всяка дума:

- ключова дума – напр. if, for, while
- идентификатор – имена на променливи, имена на типове данни, имена на функции
- константи – напр. 0, 1, 2 и т.н.
- оператори – напр. +,-,*,/,=,>,>=,<=,==
- разделители – (,),;

Резултатът от изпълнението на анализатора ще бъде двойки стойности <дума, тип>.

Примерен вход:

```
public class HelloWorld {
  public static void main() {
    System.out.println(10);
  }
}
```

Изход:

<public, ключова дума>
<class, ключова дума >
<HelloWorld, идентификатор>
<{, разделител>
<public, ключова дума>
<static, ключова дума>
…
<10, константа>
и т.н.

Анализаторът ще работи само с целочислени и символни константи.

Изход:

<int, идентификатор>
<func_change, идентификатор>
<(, разделител>
<), разделител>
<{, разделител>
<if, ключова дума>
<(, разделител>
<a, идентификатор>
<‘>‘, оператор>
и т.н.

Анализаторът ще работи само с целочислени и символни константи.

5.5.

Дадена е следната граматика:

<sentence> => <subject> <verb-phrase> <object> .
<subject> => This | Computers | I
<verb-phrase> => <adverb> <verb> | <verb>
<adverb> => never
<verb> => is | run | am | tell
<object> => the <noun> | a <noun> | <noun>
<noun> => university | world | cheese | lies

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следните низове:

This is a university.
Computers run the world.
I am the cheese.
Computers never tell lies.

3. Да се разработи програма на C++, която приема низ и извежда отговор дали този низ е в съответствие с граматиката.

Примерен вход:	Изход:
This is a university.	yes
Computers run the world.	yes
I cheese.	no

5.6.

Дадена е следната граматика:

<изречение> => <подлог> <глаголна_фраза> <допълнение> .
<подлог> => This | Computers | I
<глаголна_фраза> => <наречие> <глагол> | <глагол>
<наречие> => never
<глагол> => is | run | am | tell
<допълнение> => the <съществително_име> | a <съществително_име> | <съществително_име>
<съществително_име> => university | world | cheese | lies

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следните низове:

This is a university.
Computers run the world.
I am the cheese.
Computers never tell lies.

3. Да се разработи програма на Java, която приема низ и извежда отговор дали този низ е в съответствие с граматиката.

Примерен вход:	Изход:
This is a university.	yes
Computers run the world.	yes
I cheese.	no

4.1

Дадена е граматика за дефиниране на аритметични изрази:

p1: list => list + digit

p2: list => list - digit

p3: list => digit

p4: digit => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следните низове:

3 + 4

3 + 4 - 5

3. Да се разработи програма на C++ за анализ и изчисляване на аритметични изрази.

Примерен вход:	Изход:
2 + 3	5
2 + 3 - 4	1
2 +	Грешка

4.2

Дадена е граматика за дефиниране на аритметични изрази:

p1: list => list + digit

p2: list => list - digit

p3: list => digit

p4: digit => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи
2. Да се покаже чрез прилагане на правилата на граматиката извеждане на следните низове:

3 + 4

3 + 4 - 5

3. Да се разработи програма на Java за анализ и изчисляване на аритметични изрази.

Примерен вход:	Изход:
2 + 3	5
2 + 3 - 4	1
2 +	Грешка

4.3.

Дадена е следната граматика:

p1: <име_променлива> => <буква> (<буква> | <цифра>)*

p2: <буква> => A | B | C | D | E | F | G | H | I | | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

p3: <цифра> => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи.
2. Да се построи краен автомат (представен чрез граф) за разпознаване на име на променлива в съответствие с правило p1.
3. Да се разработи програма на C++, която получава низ и извежда резултат дали този низ е валидно име на променлива.

Примерен вход:	Изход:
ABC	valid
A1	valid
1A	invalid

4.4.

Дадена е следната граматика:

p1: <var_name> => <letter> (<letter> | <digit>)*

p2: <letter> => a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z

p3: <digit> => 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Задачи:

1. Да се представят символите на граматиката чрез синтактични графи.
2. Да се построи краен автомат (представен чрез граф) за разпознаване на var_name в съответствие с правило p1.
3. Да се разработи програма на Java, която получава низ и извежда резултат дали този низ е валидно име на променлива.

Примерен вход:	Изход:
aaa	valid
ab12	valid
1a	invalid

4.5

1. Да се състави регулярен израз, който по дадена последователност от символи (цел) и даден текст извлича от текста всички думи, които съдържат зададената цел в себе си като подниз.
2. Да се представи регулярният израз чрез краен автомат (краен ориентиран граф).
3. Да се разработи програма, която реализира задачата чрез регулярния израз.

Примерен текст:

Въпреки студеното време студентите очакват с нетърпение студентския празник.

Примерен вход:	Изход:
студ	студеното
	студентите
	студентския

4.6

1. Да се състави регулярен израз, който проверява дали дадено реално число е във формат цяла, следвана от дробна част.
2. Да се представи регулярният израз чрез краен автомат (краен ориентиран граф).
3. Да се разработи програма, която реализира задачата чрез регулярния израз.

Примерен вход:	Изход:
0	valid
66	valid
-21345.5641	valid
5.6278	valid
14.00	valid
+3	not valid
-2	not valid
29 123	not valid
01.55	not valid
13.	not valid
45,23	not valid
8e10	not valid

4.7

1. Да се състави регулярен израз, който намира в даден текстов документ всички поднизове, приличащи на e-mail адрес (последователности от символи във формат <identifier>@<host>.<domain>).
2. Да се представи регулярният израз чрез краен автомат (краен ориентиран граф).
3. Да се разработи програма, която реализира задачата чрез регулярния израз.

4.8

1. Да се състави регулярен израз, който намира в даден текстов документ всички низове, приличащи на URL адреси (поднизове във формат www.<host>.<domain> и поднизове, започващи с "http:// ").
2. Да се представи регулярният израз чрез краен автомат (краен ориентиран граф).
3. Да се разработи програма, която реализира задачата чрез регулярния израз.

4.9

1. Да се състави регулярен израз, който по даден URL адрес във формат [protocol]://[server]/[resource] извлича от него отделните му елементи – [protocol], [server] и [resource].
2. Да се представи регулярният израз чрез краен автомат (краен ориентиран граф).
3. Да се разработи програма, която реализира задачата чрез регулярния израз.

Примерен вход:	Изход:
http://cs.tu-varna.bg/news	[protocol] = “http”, [server]=”cs.tu-varna.bg”, [resource]=”/news”

4.10

Даден е речник с думи, който представлява текст във формат "дума значение", по една речникова единица на всеки ред (значението може да се състои от няколко думи). Да се разработи програма, която по дадена дума намира значението ѝ в речника. Да се използват регулярни изрази.

4.11

Да се разработи програма, която претърсва даден текст за дадена дума и намира и отпечатва всички изречения, в които тази дума се среща. Край на изречение маркират символите ".", "!" и "?". Да се използват регулярни изрази.

4.12

Да се разработи програма, която извлича от даден текст всички цели числа без знак и ги записва в масив от символни низове. Да се използват регулярни изрази.

4.13

Напишете програма, която в даден HTML документ премахва всички HTML тагове. Да се използват регулярни изрази.

Примерен входен HTML файл:

```
<html>
	<head>
		<title> Заглавие на документа </title>
	</head>
	<body>
		<p> Параграф </p>
		<a href=”http://www.tu-varna.bg”> Технически университет-Варна </a>
	</body>
</html>
```

Съдържание на файла след изпълнение на програмата:

Заглавие на документа
Параграф
Технически университет - Варна

4.14

Напишете програма, която обръща думите в дадено изречение в обратен ред. Да се използват регулярни изрази.

Примерен вход:

Навън вали силен дъжд.

Изход:

Дъжд силен вали навън.