Regular expressions. Variant 1.

1. Are the regular expressions 0(10)*1* and (01)*0(1*)* equivalent?
2. Write a regular expression for the set of all strings of 0's and 1's such as any pair of adjacent 0's appear before any pair of adjacent 1's.
3. Give English description of the language generated by the regular expression (0+01)*(0+1)*11.
4. Most languages are *case sensitive*, so keywords can be written only one way, and the regular expressions describing their lexeme is very simple. However, some languages, like SQL, are *case insensitive*, so a keyword can be written either in lowercase or in uppercase, or in any mixture of cases. Thus, the SQL keyword **SELECT** can also be written **select, Select,** or **sElEcT,** for instance. Show how to write a regular expression for a keyword in a case insensitive language. Illustrate the idea by writing the expression for "select" in SQL.
5. Write lexems and token types for the code

```
int maximum(int x,int y)
{
      return x>y?x:y;
}
```

Regular expressions. Variant 2.

1. Are the regular expressions (11+0)(11)*01 and (11)*01+01(11)*01 equivalent?
2. Write a regular expression for the set of all strings with an equal numbers of 0's and 1's such that no prefix has two more 0's then 1's, not two more 1's than 0's.
3. Give English description of the language generated by the regular expression (0+10)*1*.
4. Most languages are *case sensitive*, so keywords can be written only one way, and the regular expressions describing their lexeme is very simple. However, some languages, like SQL, are *case insensitive*, so a keyword can be written either in lowercase or in uppercase, or in any mixture of cases. Thus, the SQL keyword **SELECT** can also be written **select, Select ,** or **sElEcT,** for instance. Show how to write a regular expression for a keyword in a case insensitive language. Illustrate the idea by writing the expression for "select" in SQL.
5. Write lexems and token types for the code

```
int divide(int x,int y)
{
      return (x%y==0)?1:0;
}
```

Regular expressions. Variant 3.

1. Are the regular expressions (0+10)*1* and (0*10*)*1*.equivalent?
2. Write a regular expression for the set of all strings of 0's and 1's not containing 101 as a substring.
3. Give English description of the language generated by the regular expression (0*1*)000(0+1)*.
4. Most languages are *case sensitive*, so keywords can be written only one way, and the regular expressions describing their lexeme is very simple. However, some languages, like SQL, are *case insensitive*, so a keyword can be written either in lowercase or in uppercase, or in any mixture of cases. Thus, the SQL keyword **SELECT** can also be written **s e l e c t , S e l e c t ,** or **sElEcT,** for instance. Show how to write a regular expression for a keyword in a case insensitive language. Illustrate the idea by writing the expression for "select" in SQL.
5. Write lexems and token types for the code

```
int minimum(int x,int y)
{
      return x<y?x:y;
}
```

Regular expressions. Variant 4.

1. Are the regular expressions 0(10)*+(01)*0 and (01+10)*0 equivalent?
2. Write a regular expression for the set of all strings of 0's and 1's whose number of 0's is divisible by 3.
3. Give English description of the language generated by the regular expression (1+ε)(00*1)*0*.
4. Most languages are *case sensitive*, so keywords can be written only one way, and the regular expressions describing their lexeme is very simple. However, some languages, like SQL, are *case insensitive*, so a keyword can be written either in lowercase or in uppercase, or in any mixture of cases. Thus, the SQL keyword **SELECT** can also be written **s e l e c t , S e l e c t ,** or **sElEcT,** for instance. Show how to write a regular expression for a keyword in a case insensitive language. Illustrate the idea by writing the expression for "select" in SQL.
5. Write lexems and token types for the code

```
int even(int x,int y)
{
      return ((x*y)%2==0)?1:0;
}
```