

Практическая работа по теме: «Игра-ЗМЕЙКА»

1. Расшифруйте слова (наименование), решите ребусы:

1. 
УК = ЕЗ

2. 

3. 
+ОД 3,1,2

4. 

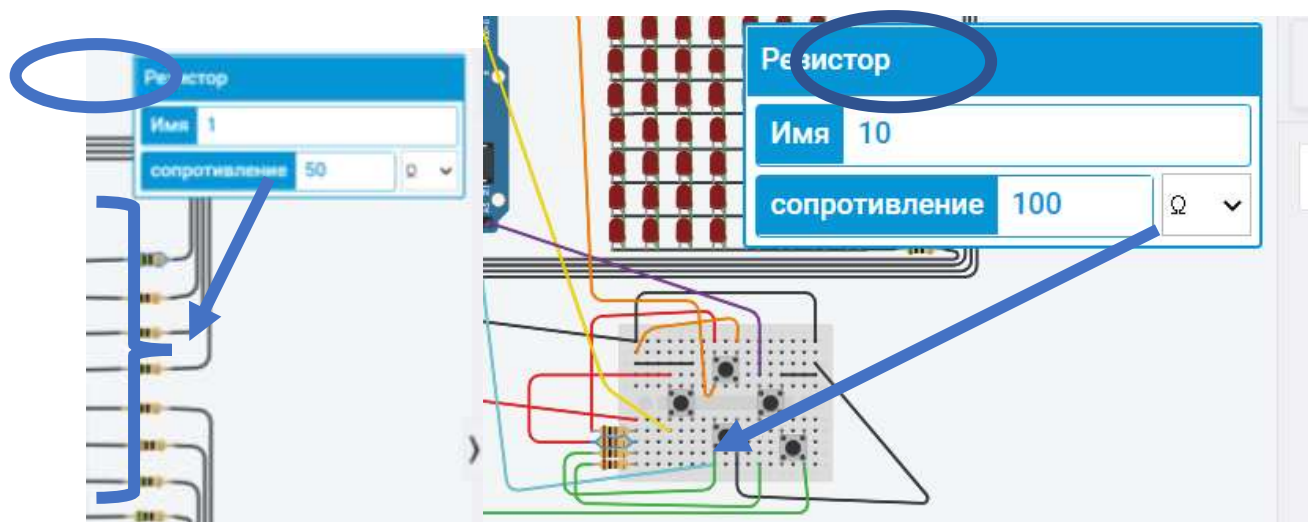
5. 

6. 
1 = ПРО

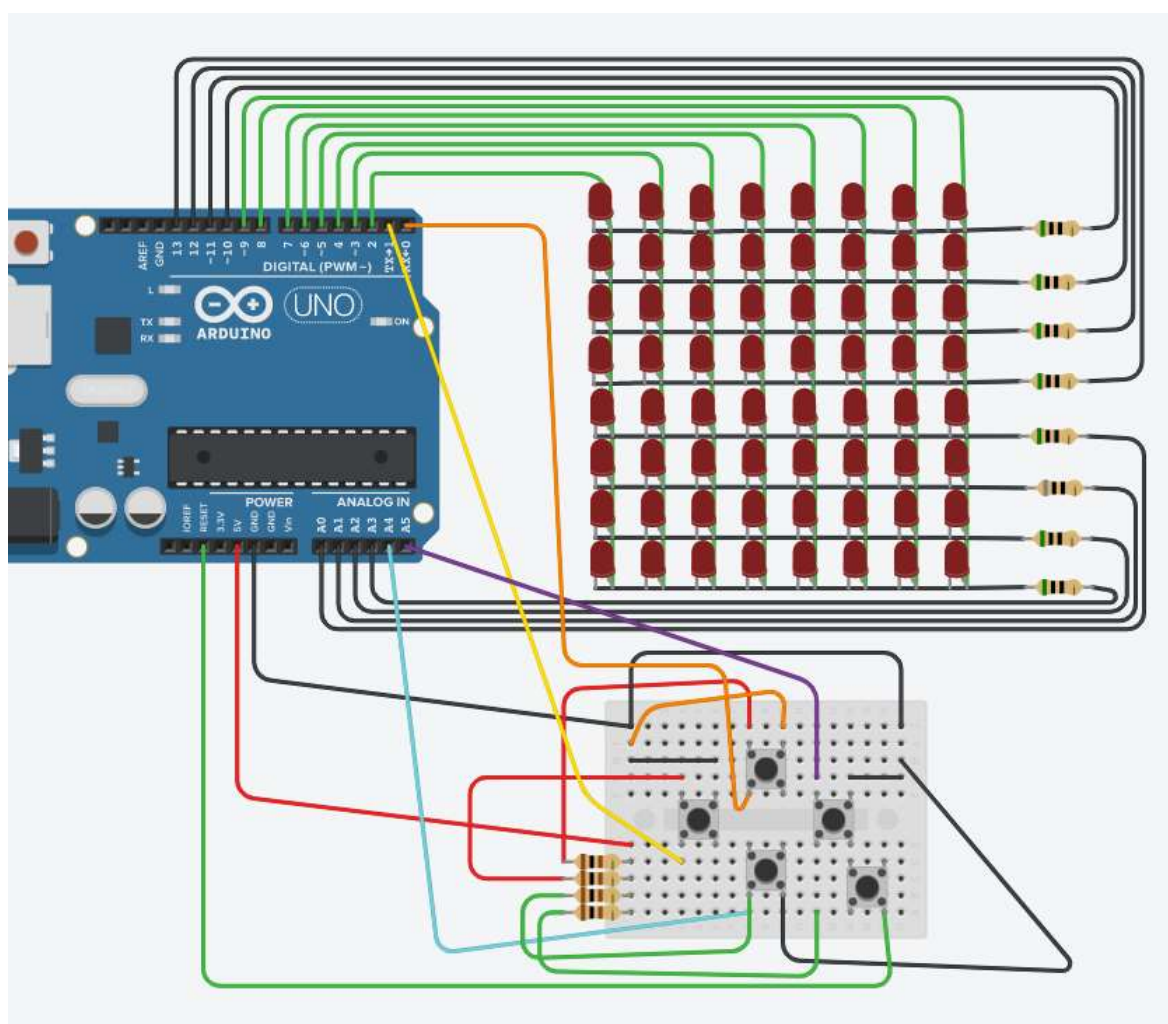
2. Заполните таблицу (наименование) используя разгаданные слова.

№ п.п.	Наименование	Количество	Внешний вид
1.			
2.			
3.			
4.			
5.			
6.			

Используйте следующие номиналы радиодеталей:



3. Соберите схему.



4. Введите код.
5. Проверьте работоспособность схемы управляя кнопками.

```

// C++ code
//
int snakelen = 3;
int pos[64][2] = {{4,4},{4,5},{4,6}};
int pre = 3;
int dir = 4;
int apple[2] = {1,1};
int g = 0;
int run = 0;
void setup()
{
  for (int j=2; j<18; j++)
  {
    pinMode(j,OUTPUT);

    pinMode(0,INPUT);
    pinMode(1,INPUT);
    pinMode(18,INPUT);
    pinMode(19,INPUT);
  }
}
void render(int a[63][2])
{
  for(int i=2; i<=9; i++)
  {
    digitalWrite(i,HIGH);
    for(int j=0; j<=snakelen; j++)
    {
      if(i==a[j][0]+2)
      {
        digitalWrite(a[j][1]+10,LOW);
      }
      else
      {
        continue;
      }
    }
    delay(2);
    for(int j=10; j<=17; j++)
    {
      digitalWrite(j,HIGH);
    }
    for(int i=2; i<=9; i++)
    {
      digitalWrite(i,LOW);
    }
  }
}
int bound(int a)
{
  if(a<0)
  {
    a = 8+a;
  }
  else
  {
    a = a;
  }
}

return a;
}
void newapple()
{
  apple[0] = bound(rand()%8);
  apple[1] = bound(rand()%8);
}
void loop()
{
  while(run<1)
  {
    if(digitalRead(19)==0)
    {
      dir=1;
    }
    else if(digitalRead(1)==0)
    {
      dir=3;
    }
    else if(digitalRead(18)==0)
    {
      dir=2;
    }
    else if(digitalRead(0)==0)

```

```

{
    dir =4;
}

if(g == 0)
{
    for(int i = snakelen-1; i>=1 ; i--)
    {
        pos[i][0] = bound(pos[i-1][0]%8);
        pos[i][1] = bound(pos[i-1][1]%8);
    }
}
else if(g ==1)
{
    snakelen += 1;
    for(int i = snakelen-1; i>=1 ; i--)
    {
        pos[i][0] = bound(pos[i-1][0]%8);
        pos[i][1] = bound(pos[i-1][1]%8);
    }
    g = 0;
}
switch(dir)
{
case 1:
    if (pre != 3)
    {
        pos[0][0] = bound((pos[0][0]+1)%8); // +x right
        pre = dir;
    }
    else
    {
        pos[0][0] = bound((pos[0][0]-1)%8); // -x left
    }
    break;
case 3:
    if (pre != 1)
    {
        pos[0][0] = bound((pos[0][0]-1)%8); // -x left
        pre = dir;
    }
    else
    {
        pos[0][0] = bound((pos[0][0]+1)%8); // +x right
    }
    break;
case 2:
    if (pre != 4)
    {
        pos[0][1] = bound((pos[0][1]+1)%8); // +y downwards
        pre = dir;
    }
    else
    {
        pos[0][1] = bound((pos[0][1]-1)%8); // -y upwards
    }
    break;
case 4:
    if (pre != 2)
    {
        pos[0][1] = bound((pos[0][1]-1)%8); // -y upwards
        pre = dir;
    }
    else
    {
        pos[0][1] = bound((pos[0][1]+1)%8); // +y downwards
    }
    break;
}

for( int i =0; i<=snakelen-1;i++)
{
    if(pos[0][0] ==apple[0] && pos[0][1] ==apple[1])
    {
        g = 1;
        newapple();
    }
}
for( int i =1; i<=snakelen-1;i++)
{
    if(pos[0][0] ==pos[i][0] && pos[0][1] ==pos[i][1])
    {
        run = 2;
    }
}

```

```

    }
  }
  pos[snakelen][0] = apple[0];
  pos[snakelen][1] = apple[1];
  render(pos);
  pos[snakelen][0] = 0;
  pos[snakelen][1] = 0;
  delay(200); // Delay a little bit to improve simulation performance
}

  for(int j=10; j<=17; j++)
  {
    digitalWrite(j,HIGH);
  }

  for(int i=2; i<=9; i++)
  {
    digitalWrite(i,LOW);
  }
  delay(1000);
}

```