Task3 Key Solution

```
class Contact
{
    string _name;
    string _address;
    List<int> _phoneNums = new List<int>();
    string _type;

    3 references
    public string Name
    {
        get { return _name; }
    }

    1 reference
    public List<int> Phone
    {
        set { _phoneNums = value; }
        get { return _phoneNums; }

    }
    2 references
    public string Type
    {
        get { return _type; }
    }

    1 reference
    public string Address
    {
        get { return _address; }
    }
    public Contact ()
    {
        _name = "";
        _address = "";
        _type = "";
    }

    1 reference
    public Contact(string n, string a, List<int> p, string t)
    {
        _name = n;
        _address = a;
        _phoneNums = p;
        _type = t;
    }

    3 references
    public void PrintInfo()
    {
        Console.Write($"Name: {_name}\tAddress: {_address}\tType:{_type}\t");

        Console.Write("Phone numbers: ");
        foreach (int phone in _phoneNums)
            Console.Write($"{phone}\t");

        Console.WriteLine();
    }
```

```
    public  bool IsExist(int num)
    {
        return _phoneNums.Contains(num);
    }

    2 references
    public int PhoneCount()
    {
        return _phoneNums.Count;
    }
}

static void Main(string[] args)
{
    List<Contact> AdressBook = new List<Contact>();

    while(true)
    {
        Console.WriteLine("1.   Add a new contact ");
        Console.WriteLine("2.   Remove a contact ");
        Console.WriteLine("3.   Add Phone number ");
        Console.WriteLine("4.   Count the total number of phones ");
        Console.WriteLine("5.   Count the number of phone numbers--> Type ");
        Console.WriteLine("6.   Count the number of friends' contacts in a specific city");
        Console.WriteLine("7.   Print person's info");
        Console.WriteLine("8.   Search for a phone");
        Console.WriteLine("9.   Print all info in the address book");
        Console.WriteLine("10.  Exit");

        int choice = int.Parse(Console.ReadLine());
        switch(choice)
        {
            case 1:
                {
                    Console.WriteLine("insert name");
                    string name = Console.ReadLine();
                    Console.WriteLine("insert address");
                    string address = Console.ReadLine();
                    Console.WriteLine("insert type");
                    string type = Console.ReadLine();

                    List<int> nums = new List<int>();
                    int num;
                    char ch = 'Y';
                    while (ch == 'Y')
                    {
                        Console.WriteLine("insert phone num");
                        num = int.Parse(Console.ReadLine());
                        if (num.ToString().Length+1 == 10)
                            nums.Add(num);

                        Console.WriteLine("another phone Y/ N");
                        ch = char.Parse(Console.ReadLine());
                    }

                    Contact contact = new Contact(name, address, nums, type);
                    AdressBook.Add(contact);
                }
                break;
```

```
        case 2:
            {
                Console.WriteLine("insert name");
                string name = Console.ReadLine();
                AdressBook.RemoveAll(cont => cont.Name == name);

            }
            break;
        case 3:
            {
                Console.WriteLine("insert name");
                string name = Console.ReadLine();
                Console.WriteLine("insert phone num");
                int num = int.Parse(Console.ReadLine());
                if (num.ToString().Length+1 == 10)
                {
                    foreach (Contact c in AdressBook)
                    {
                        if (c.Name == name)
                            c.Phone.Add(num);
                    }
                }
                else Console.WriteLine("wrong num entered");

            }
            break;
        case 4:
            {
                int sum = 0;
                foreach (Contact c in AdressBook)
                {
                    sum += c.PhoneCount();
                }
                Console.WriteLine("total num of phones = {0}", sum);
            }
                break;
        case 5:
            {
                Console.WriteLine("insert type");
                string type = Console.ReadLine();

                int sum = 0;
                foreach (Contact c in AdressBook)
                {
                    if(c.Type == type)
                        sum += c.PhoneCount();
                }
                Console.WriteLine("total num of phones = {0}", sum);
            }
            break;
        case 6:
            {
                Console.WriteLine("insert address");
                string address = Console.ReadLine();
                int sum = 0;
                foreach (Contact c in AdressBook)
                {
                    if (c.Type == "friend" && c.Address == address)
                        sum++;
                }
                Console.WriteLine("total num of friends = {0}", sum);

            }
            break;
```

```
            case 7:
                {
                    Console.WriteLine("insert name");
                    string name = Console.ReadLine();

                    foreach (Contact c in AdressBook)
                    {
                        if (c.Name == name)
                            c.PrintInfo();
                    }

                }
                break;
            case 8:
                {
                    Console.WriteLine("insert phone num");
                    int num = int.Parse(Console.ReadLine());
                    bool found = false;
                    foreach (Contact c in AdressBook)
                    {
                        if (c.IsExist(num))
                        {
                            c.PrintInfo();
                            found = true;
                        }
                    }

                    if(!found) Console.WriteLine("phone num not found");

                }
                break;
            case 9:
                {
                    foreach (Contact c in AdressBook)
                    {
                        c.PrintInfo();
                    }
                }
                break;
            case 10:
                {
                    Environment.Exit(0);
                }
                break;
        }
    }
}
```

Task4 Key Solution

```
public partial class Form1 : Form
{
    List<string> contacts = new List<string>();
    int indx;
    public Form1()
    {
        InitializeComponent();
        contacts.Add("Name\t\tAddress\t\tType\t\tMobile\r\n");
        contacts.Add("Ahmad\t\tIrbid\t\tFamily\t\t0777777777\r\n");
        contacts.Add("Mohammad\tAmman\t\tFriend\t\t0771111111\r\n");
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        updateRich();
    }

    private void btnSearch_Click(object sender, EventArgs e)
    {
        foreach (string s in contacts)
        {
            if(s.Contains(txtSearch.Text))
            {
                richTextBox1.Clear();
                richTextBox1.Text = s;

                btnAdd.Enabled = false;
                gBoxContactInfo.Visible = true;
                btnDelete.Visible = btnUpdate.Visible = true;
                btnOk.Visible = false;

                char[] separators = new char[] { '\t' };
                string[] sub = s.Split(separators, StringSplitOptions.RemoveEmptyEntries);
                txtName.Text = sub[0];
                txtCity.Text = sub[1];
                cboType.SelectedItem = (object)sub[2];
                txtNum.Text = sub[3];

                indx=contacts.IndexOf(s);
            }
        }
    }

    private void btnClear_Click(object sender, EventArgs e)
    {
        txtSearch.Clear();
        updateRich();
    }
```

```
    private void btnDelete_Click(object sender, EventArgs e)
    {
        //foreach (string s in contacts)
        //{
        //    if (s.Contains(txtName.Text))
        //    {
        //        contacts.Remove(s);
        //        MessageBox.Show("Contact removed successfully");
        //    }
        //}

        contacts.RemoveAll(s => s.Contains(txtName.Text));
        MessageBox.Show("Contact removed successfully");

        updateRich();
    }

    private void btnAdd_Click(object sender, EventArgs e)
    {
        txtName.Clear();
        txtCity.Clear();
        txtNum.Clear();
        cboType.Text = "";
        gBoxContactInfo.Visible = true;
        btnDelete.Visible = btnUpdate.Visible = false;
        btnOk.Visible = true;
    }

    private void btnOk_Click(object sender, EventArgs e)
    {
        contacts.Add($"{txtName.Text}\t{txtCity.Text}\t\t{cboType.SelectedItem.ToString()}\t\t{txtNum.Text}\r\n");
        MessageBox.Show("Contact Added successfully");

        updateRich();
    }

    private void btnUpdate_Click(object sender, EventArgs e)
    {
        contacts.RemoveAt(indx);
        contacts.Insert(indx, $"{txtName.Text}\t{txtCity.Text}\t\t{cboType.SelectedItem.ToString()}\t\t{txtNum.Text}\r\n");
        MessageBox.Show("Contact Updated successfully");


        updateRich();
    }

    private void updateRich()
    {
        richTextBox1.Clear();

        foreach (string s in contacts)
        {
            richTextBox1.Text += s;
        }

        btnAdd.Enabled = true;
        gBoxContactInfo.Visible = false;
        btnOk.Visible = btnDelete.Visible = btnUpdate.Visible = true;
    }
}
```

Task5 Key Solution

```
public partial class Form1 : Form
{
    int num;
    List<string> Names = new List<string>();
    List<int> Prices = new List<int>();
    List<int> Amounts = new List<int>() { 0,0,0,0 };
    public Form1()
    {
        InitializeComponent();
        num = 1;
    }

    private void redToolStripMenuItem_Click(object sender, EventArgs e)
    {
        richTextBox1.ForeColor = Color.Red;
    }
    private void blackToolStripMenuItem_Click(object sender, EventArgs e)
    {
        richTextBox1.ForeColor = Color.Black;
    }
    private void greenToolStripMenuItem_Click(object sender, EventArgs e)
    {
        richTextBox1.BackColor = Color.Green;
    }
    private void whiteToolStripMenuItem_Click(object sender, EventArgs e)
    {
        richTextBox1.BackColor = Color.White;
    }
    private void consolasToolStripMenuItem_Click(object sender, EventArgs e)
    {
        richTextBox1.Font = new Font("Consolas", richTextBox1.Font.Size, richTextBox1.Font.Style);
    }
    private void boldToolStripMenuItem_Click(object sender, EventArgs e)
    {
        richTextBox1.Font = new Font(richTextBox1.Font.FontFamily, richTextBox1.Font.Size, FontStyle.Bold);
    }
    private void regularToolStripMenuItem_Click(object sender, EventArgs e)
    {
        richTextBox1.Font = new Font(richTextBox1.Font.FontFamily, richTextBox1.Font.Size, FontStyle.Regular);
    }
```

```
private void openToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        StreamReader reader = new StreamReader(openFileDialog1.FileName);

        string line = reader.ReadLine(); //1st line
        string[] sub = line.Split(' ');

        foreach (string s in sub)
            Names.Add(s);


        line = reader.ReadLine(); //2nd line
        sub = line.Split(' ');

        foreach (string s in sub)
            Prices.Add(Convert.ToInt32(s));

        line = reader.ReadLine(); //3rd line and on

        while (line != null)
        {
            sub = line.Split(' ');

            int indx = Names.IndexOf(sub[0]);
            Amounts[indx] += Convert.ToInt32(sub[1]);

            line = reader.ReadLine();
        }

        reader.Close();

        richTextBox1.Clear();
        richTextBox1.Text = "Item\t\t Price\t\tAmount\n";
        int totalPrice = 0;

        for (int i=0;i<Names.Count;i++)
        {
            richTextBox1.Text += $"{Names[i]}\t\t{Prices[i]}\t\t{Amounts[i]}\n";
            totalPrice += Amounts[i]* Prices[i];
        }
        richTextBox1.Text += $"\n\n\nTotal Sales = {totalPrice} JD\n";
    }
}
```

```
private void saveToolStripMenuItem_Click(object sender, EventArgs e)
{
    if(saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        StreamWriter writer = new StreamWriter(saveFileDialog1.FileName);

        writer.WriteLine(richTextBox1.Text);

        writer.Close();
    }
}


private void timer1_Tick(object sender, EventArgs e)
{
    Random rand = new Random();
    num = rand.Next(1, 5);
    setImage(num);
}

private void Previous_Click(object sender, EventArgs e)
{
    timer1.Stop();

    num--;
    if (num < 1)
        num = 4;

    setImage(num);

    timer1.Start();
}
```

```
private void btnNext_Click(object sender, EventArgs e)
{
    timer1.Stop();

    num++;
    if (num > 4)
        num = 1;

    setImage(num);

    timer1.Start();
}

private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

```
    private void setImage(int num)
    {
        switch (num)
        {
            case 1:
                pictureBox1.Image = Csharp.Properties.Resources.Table;
                break;
            case 2:
                pictureBox1.Image = Csharp.Properties.Resources.Sofa;
                break;
            case 3:
                pictureBox1.Image = Csharp.Properties.Resources.Cupboard;
                break;
            case 4:
                pictureBox1.Image = Csharp.Properties.Resources.Bed;
                break;
        }
    }
}
```

Task6 Key Solution

```
class Course
{
    List<string> names;
    List<int> marks;

    public Course()
    {
        names = new List<string>();
        marks = new List<int>();
    }

    public void AddStudent(string name, int mark)
    {
        if(!names.Contains(name))
        {
            names.Add(name);
            marks.Add(mark);
        }
    }
    public string ViewCourseInfo()
    {
        string info = "Student Name \t Student Mark\n";
        for (int i=0; i<names.Count;i++)
        {
            info += names[i] + "\t" + marks[i] + "\n";
        }
        return info;
    }
    public int Max(){ return marks.Max();}
    public int Min() { return marks.Min(); }
    public double Avg() { return marks.Average(); }
```

```
    public int Passed()
    {
        int count = 0;
        foreach (int m in marks)
            if (m >= 50) count++;

        return count;
    }

    public int Faild()
    {
        int count = 0;
        foreach (int m in marks)
            if (m < 50) count++;

        return count;
    }
}

public partial class Form1 : Form
{
    Course myCourse;
    public Form1()
    {
        InitializeComponent();
        myCourse = new Course();

    }

    private void btn_Insert_Click(object sender, EventArgs e)
    {
        myCourse.AddStudent(txt_Name.Text, Convert.ToInt32(txt_Mark.Text));

        richTextBox1.Text = "";
        richTextBox1.Text += myCourse.ViewCourseInfo();

        txt_Max.Text = myCourse.Max().ToString();
        txt_Min.Text = myCourse.Min().ToString();
        txt_Avg.Text = myCourse.Avg().ToString();
        txt_Pass.Text = myCourse.Passed().ToString();
        txt_Fail.Text = myCourse.Faild().ToString();
    }
}
```

Task7 Key Solution

```
public partial class Form1 : Form
{
    int lives;
    int time;
    Random rand = new Random();
    public Form1()
    {
        InitializeComponent();
        lives = 5;
        time = 0;
    }

    private void Form1_Load(object sender, EventArgs e)
    {

        label1.Text = "No. Lives = "+lives.ToString();
        pictureBox1.Location = new Point(rand.Next(0, this.ClientRectangle.Width- pictureBox1.Size.Width), rand.Next(0, this.ClientRectangle.Height));
        pictureBox2.Location = new Point(rand.Next(0, this.ClientRectangle.Width - pictureBox2.Size.Width), rand.Next(0, this.ClientRectangle.Height));
        pictureBox3.Location = new Point(rand.Next(0, this.ClientRectangle.Width - pictureBox3.Size.Width), rand.Next(0, this.ClientRectangle.Height));
        pictureBox4.Location = new Point(rand.Next(0, this.ClientRectangle.Width - pictureBox4.Size.Width), rand.Next(0, this.ClientRectangle.Height));
        pictureBox5.Location = new Point(rand.Next(0, this.ClientRectangle.Width - pictureBox5.Size.Width), rand.Next(0, this.ClientRectangle.Height));
    }
```

```
private void timer1_Tick(object sender, EventArgs e)
{
    //move rocks down each tick
    pictureBox1.Location = new Point(pictureBox1.Location.X, pictureBox1.Location.Y + 10);
    pictureBox2.Location = new Point(pictureBox2.Location.X, pictureBox2.Location.Y + 10);
    pictureBox3.Location = new Point(pictureBox3.Location.X, pictureBox3.Location.Y + 10);
    pictureBox4.Location = new Point(pictureBox4.Location.X, pictureBox4.Location.Y + 10);
    pictureBox5.Location = new Point(pictureBox5.Location.X, pictureBox5.Location.Y + 10);

    //if a rock reaches the bottom of the form relocate it in random location // Extra-- Not required in the task
    if(reachedBoarders(pictureBox1.Location.Y))
        pictureBox1.Location = new Point(rand.Next(0, this.ClientRectangle.Width - pictureBox1.Size.Width), rand.Next(0, this.ClientRectangle.Height));

    if (reachedBoarders(pictureBox2.Location.Y))
        pictureBox2.Location = new Point(rand.Next(0, this.ClientRectangle.Width - pictureBox2.Size.Width), rand.Next(0, this.ClientRectangle.Height));

    if (reachedBoarders(pictureBox3.Location.Y))
        pictureBox3.Location = new Point(rand.Next(0, this.ClientRectangle.Width - pictureBox3.Size.Width), rand.Next(0, this.ClientRectangle.Height));

    if (reachedBoarders(pictureBox4.Location.Y))
        pictureBox4.Location = new Point(rand.Next(0, this.ClientRectangle.Width - pictureBox4.Size.Width), rand.Next(0, this.ClientRectangle.Height));

    if (reachedBoarders(pictureBox5.Location.Y))
        pictureBox5.Location = new Point(rand.Next(0, this.ClientRectangle.Width - pictureBox5.Size.Width), rand.Next(0, this.ClientRectangle.Height));

    //to calculate the two min
    time++;

    //check for overlab
    if (IsOverlab2()) lives--;
    label1.Text = "No. Lives = " + lives.ToString();

    //check for win or lose
    if (lives == 0)
    {
        timer1.Stop();
        MessageBox.Show("Game Over!");
        Application.Exit();
    }
    else if(time == 60)
    {
        timer1.Stop();
        MessageBox.Show("You Won!");
        Application.Exit();
    }


}
```

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Left)
        pictureBox6.Location = new Point(pictureBox6.Location.X - 10, pictureBox6.Location.Y);
    else if (e.KeyCode == Keys.Right)
        pictureBox6.Location = new Point(pictureBox6.Location.X + 10, pictureBox6.Location.Y);

    else if (e.KeyCode == Keys.Up)
        pictureBox6.Location = new Point(pictureBox6.Location.X , pictureBox6.Location.Y-10);
    else if (e.KeyCode == Keys.Down)
        pictureBox6.Location = new Point(pictureBox6.Location.X, pictureBox6.Location.Y + 10);

    //check for overlab
    if (IsOverlab2()) lives--;
    label1.Text = "No. Lives = " + lives.ToString();

    //check for win or lose
    if (lives == 0)
    {
        timer1.Stop();
        MessageBox.Show("Game Over!");
        Application.Exit();
    }
    else if (time == 60)
    {
        timer1.Stop();
        MessageBox.Show("You Won!");
        Application.Exit();
    }
}
```

```
    //functions to check overlab
    private bool IsOverlab2()
    {
        if (pictureBox6.Bounds.IntersectsWith(pictureBox1.Bounds)) return true;
        else if (pictureBox6.Bounds.IntersectsWith(pictureBox2.Bounds)) return true;
        else if (pictureBox6.Bounds.IntersectsWith(pictureBox3.Bounds)) return true;
        else if (pictureBox6.Bounds.IntersectsWith(pictureBox4.Bounds)) return true;
        else if (pictureBox6.Bounds.IntersectsWith(pictureBox5.Bounds)) return true;
        else return false;
    }
    private bool IsOverlab()
    {
        if (pictureBox6.Location.X >= pictureBox1.Location.X && pictureBox6.Location.X <= pictureBox1.Location.X + pictureBox1.Size.Width)

            if (pictureBox6.Location.Y >= pictureBox1.Location.Y && pictureBox6.Location.Y <= pictureBox1.Location.Y + pictureBox1.Size.Height)
                return true;
        if (pictureBox6.Location.X >= pictureBox2.Location.X && pictureBox6.Location.X <= pictureBox2.Location.X + pictureBox2.Size.Width)

            if (pictureBox6.Location.Y >= pictureBox2.Location.Y && pictureBox6.Location.Y <= pictureBox2.Location.Y + pictureBox2.Size.Height)
                return true;

        if (pictureBox6.Location.X >= pictureBox3.Location.X && pictureBox6.Location.X <= pictureBox3.Location.X + pictureBox3.Size.Width)
            if (pictureBox6.Location.Y >= pictureBox3.Location.Y && pictureBox6.Location.Y <= pictureBox3.Location.Y + pictureBox3.Size.Height)
                return true;

        if (pictureBox6.Location.X >= pictureBox4.Location.X && pictureBox6.Location.X <= pictureBox4.Location.X + pictureBox4.Size.Width)

            if (pictureBox6.Location.Y >= pictureBox4.Location.Y && pictureBox6.Location.Y <= pictureBox4.Location.Y + pictureBox4.Size.Height)
                return true;
        if (pictureBox6.Location.X >= pictureBox5.Location.X && pictureBox6.Location.X <= pictureBox5.Location.X + pictureBox5.Size.Width)

            if (pictureBox6.Location.Y >= pictureBox5.Location.Y && pictureBox6.Location.Y <= pictureBox5.Location.Y + pictureBox5.Size.Height)
                return true;
        return false;
    }


    //function to check if point exceeds the bottom of the form
    private bool reachedBoarders(int p)
    {
        return (p > this.ClientRectangle.Height);
    }
}
```

Task8 Key Solution

```
public partial class Form1 : Form
{
    int flag;
    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {

    }

    private void btn_Line_Click(object sender, EventArgs e)
    {
        flag = 1;
        panel1.Invalidate();
    }

   private void button2_Click(object sender, EventArgs e)
    {
        flag = 2;
        panel1.Invalidate();

    }
```

```
private void btn_Pie_Click(object sender, EventArgs e)
{
    flag = 3;
    panel1.Invalidate();

}
```

```
private void panel1_Paint(object sender, PaintEventArgs e)
{
   if (flag == 1)
   {
       Graphics g = e.Graphics;
       Pen p = new Pen(Color.Black, 3);

       g.Clear(panel1.BackColor);

       //x and y accesses
       g.DrawLine(p, 10, 20, 10, 270); //(p, 327, 46, 327, 300);
       g.DrawLine(p, 10, 270, 460, 270);

       //calculate delta
       int sum = Int32.Parse(txt_excellent.Text) + Int32.Parse(txt_VG.Text) +
           Int32.Parse(txt_G.Text) + Int32.Parse(txt_Acc.Text) + Int32.Parse(txt_Week.Text) + Int32.Parse(txt_Fail.Text);

       int delta = 250 / sum;

       //calculate height

       int h1 = (delta * Int32.Parse(txt_excellent.Text));
       int h2 = (delta * Int32.Parse(txt_VG.Text));
       int h3 = (delta * Int32.Parse(txt_G.Text));
       int h4 = (delta * Int32.Parse(txt_Acc.Text));
       int h5 = (delta * Int32.Parse(txt_Week.Text));
       int h6 = (delta * Int32.Parse(txt_Fail.Text));

       //find points
       Point p1 = new Point(40, 270 - h1);
       Point p2 = new Point(120, 270 - h2);
       Point p3 = new Point(200, 270 - h3);
       Point p4 = new Point(280, 270 - h4);
       Point p5 = new Point(360, 270 - h5);
       Point p6 = new Point(440, 270 - h6);

      //draw line
      Pen myPen = new Pen(Color.Red);
      g.DrawLine(myPen, p1, p2);
      g.DrawLine(myPen, p2, p3);
      g.DrawLine(myPen, p3, p4);
      g.DrawLine(myPen, p4, p5);
      g.DrawLine(myPen, p5, p6);

      // g.DrawRectangle(p,40, 270 - h, 20,h);

     // g.Dispose();
      p.Dispose();
      myPen.Dispose();
   }
```

```
else if (flag == 2)
{
    Graphics g = e.Graphics;
    Pen p = new Pen(Color.Black, 3);

    g.Clear(panel1.BackColor);

    //x and y accesses
    g.DrawLine(p, 10, 20, 10, 270); //(p, 327, 46, 327, 300);
    g.DrawLine(p, 10, 270, 460, 270);

    //calculate delta
    int sum = Int32.Parse(txt_excellent.Text) + Int32.Parse(txt_VG.Text) +
        Int32.Parse(txt_G.Text) + Int32.Parse(txt_Acc.Text) + Int32.Parse(txt_Week.Text) + Int32.Parse(txt_Fail.Text);

    int delta = 250 / sum;

    //calculate height

    int h1 = (delta * Int32.Parse(txt_excellent.Text));
    int h2 = (delta * Int32.Parse(txt_VG.Text));
    int h3 = (delta * Int32.Parse(txt_G.Text));
    int h4 = (delta * Int32.Parse(txt_Acc.Text));
    int h5 = (delta * Int32.Parse(txt_Week.Text));
    int h6 = (delta * Int32.Parse(txt_Fail.Text));

    //find points
    Point p1 = new Point(40, 270 - h1);
    Point p2 = new Point(120, 270 - h2);
    Point p3 = new Point(200, 270 - h3);
    Point p4 = new Point(280, 270 - h4);
    Point p5 = new Point(360, 270 - h5);
    Point p6 = new Point(440, 270 - h6);

    //draw Rectangle

    SolidBrush myBrush = new SolidBrush(Color.Red);

    g.FillRectangle(myBrush, p1.X, p1.Y, 20, h1);
    g.FillRectangle(myBrush, p2.X, p2.Y, 20, h2);
    g.FillRectangle(myBrush, p3.X, p3.Y, 20, h3);
    g.FillRectangle(myBrush, p4.X, p4.Y, 20, h4);
    g.FillRectangle(myBrush, p5.X, p5.Y, 20, h5);
    g.FillRectangle(myBrush, p6.X, p6.Y, 20, h6);

    p.Dispose();
    myBrush.Dispose();
}
else if (flag == 3)
{
    Graphics g = e.Graphics;
    g.Clear(panel1.BackColor);


     int i1 = Int32.Parse(txt_excellent.Text);
     int i2 = Int32.Parse(txt_VG.Text);
     int i3 = Int32.Parse(txt_G.Text);
     int i4 = Int32.Parse(txt_Acc.Text);
     int i5 = Int32.Parse(txt_Week.Text);
     int i6 = Int32.Parse(txt_Fail.Text);

     float sum = i1 + i2 + i3 + i4 + i5;
     float delta = 360 / sum;
```

```
            float deg1 = i1 * delta;
            float deg2 = i2 * delta;
            float deg3 = i3 * delta;
            float deg4 = i4 * delta;
            float deg5 = i5 * delta;
            float deg6 = i6 * delta;

            Rectangle rect = new Rectangle(100, 50, 200, 200);


            Brush brush1 = new SolidBrush(Color.Red);
            Brush brush2 = new SolidBrush(Color.Blue);
            Brush brush3 = new SolidBrush(Color.Maroon);
            Brush brush4 = new SolidBrush(Color.Navy);
            Brush brush5 = new SolidBrush(Color.YellowGreen);
            Brush brush6 = new SolidBrush(Color.Green);

            g.FillPie(brush1, rect, 0, deg1);
            g.FillPie(brush2, rect, deg1, deg2);
            g.FillPie(brush3, rect, deg1 + deg2, deg3);
            g.FillPie(brush4, rect, deg1 + deg2 + deg3, deg4);
            g.FillPie(brush5, rect, deg1 + deg2 + deg3 + deg4, deg5);
            g.FillPie(brush6, rect, deg1 + deg2 + deg3 + deg4 + deg5, deg6);

           // g.Dispose();
            brush1.Dispose();
            brush2.Dispose();
            brush3.Dispose();
            brush4.Dispose();
            brush5.Dispose();
            brush6.Dispose();
        }
    }
}
```

Task9 Key Solution

```
class Employee
{
    protected string name;
    protected int id;
    protected double salary;

    public Employee (string name, int id, double salary)
    {
        this.name = name;
        this.id = id;
        this.salary = salary;
    }

    public virtual double CalculateSalary()
    { return salary; }

    public virtual string PrintInfo()
    {
        return $"Employee Name: {name}, ID: {id}, Salary: {CalculateSalary()}";
    }
}
```

```
class Administrative : Employee
{
    int extraHours;

    public Administrative(string name, int id, double salary, int hours): base(name,id,salary)
    {
        extraHours = hours;
    }
    public override double CalculateSalary()
    {
        return base.CalculateSalary() + extraHours * 3;
    }

    public override string PrintInfo()
    {

        return "Administrative " + base.PrintInfo() + $" Extra Hours = {extraHours}";
    }
}


class Academic : Employee
{
    int numOfCourses;

    public Academic(string name, int id, double salary, int num) : base(name, id, salary)
    {
        numOfCourses = num;
    }
    public override double CalculateSalary()
    {
       return base.CalculateSalary() * numOfCourses;
    }

    public override string PrintInfo()
    {
        return "Academic " + base.PrintInfo() + $" No. Courses = {numOfCourses}";
    }
}

class Department
{
    List<Employee> employees;

    public Department ()
    {
        employees = new List<Employee>();
    }

    public void AddEmployee(Employee e)
    {
        employees.Add(e);
    }

    public string PrintEmployeesInfo()
    {
        string info = "";
        foreach(Employee e in employees)
        {
           info += e.PrintInfo() +"\n";
        }

        return info;
    }
}
```

```
public partial class Form1 : Form
{
    Department myDept;
    public Form1()
    {
        InitializeComponent();
        myDept = new Department();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        groupBox1.Visible = false;
    }

    private void btn_new_Click(object sender, EventArgs e)
    {
        groupBox1.Visible = true;
        lbl_extra_num.Visible = txt_extra_num.Visible = false;
    }

    private void btn_add_Click(object sender, EventArgs e)
    {
        string name = txt_name.Text;
        int id = Convert.ToInt32(txt_id.Text);
        double bSalary = Convert.ToDouble(txt_bSalary.Text);
        string type = cbo_type.SelectedItem.ToString();
        int extra_num = Convert.ToInt32(txt_extra_num.Text);
        Employee emp;

        if(type == "Administrative")
        {
            emp = new Administrative(name, id, bSalary, extra_num);
        }
        else
        {
            emp = new Academic(name, id, bSalary, extra_num);
        }

        myDept.AddEmployee(emp);
        MessageBox.Show("Employee Added Successfully!");
        txt_name.Text = txt_id.Text = txt_bSalary.Text = txt_extra_num.Text = "";
        cbo_type.Text = "";
        lbl_extra_num.Visible = txt_extra_num.Visible = false;
    }


    private void cbo_type_SelectedIndexChanged(object sender, EventArgs e)
    {
        string type = cbo_type.SelectedItem.ToString();
        if (type == "Administrative")
        {
            lbl_extra_num.Text = "Extra Hours: ";

        }
        else
        {
            lbl_extra_num.Text = "No. Courses: ";
        }
        lbl_extra_num.Visible = txt_extra_num.Visible = true;
    }

    private void btn_print_Click(object sender, EventArgs e)
    {
        richTextBox1.Text = myDept.PrintEmployeesInfo();
    }
}
```