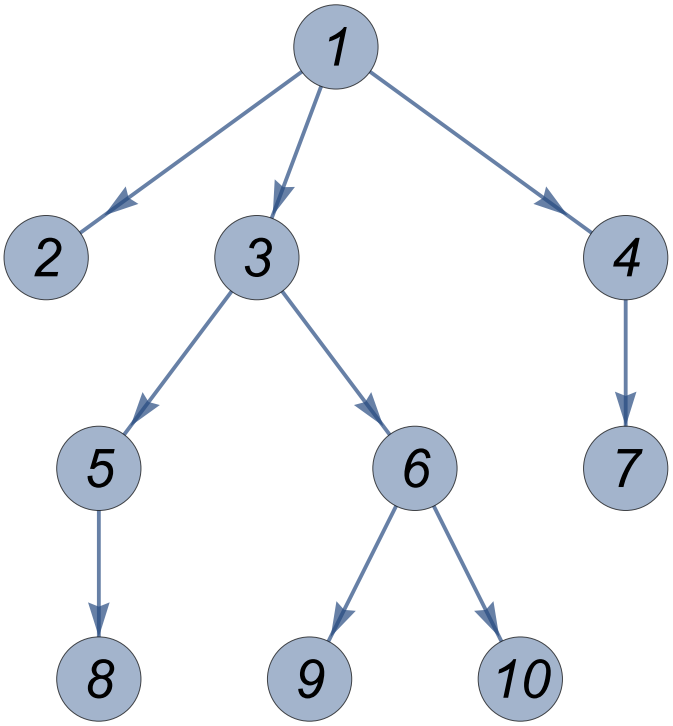
In[1]:=
```
e = {1 ↔ 2, 1 ↔ 3, 1 ↔ 4, 3 ↔ 5, 3 ↔ 6, 5 ↔ 8, 6 ↔ 9, 4 ↔ 7, 6 ↔ 10};
root = 1;
g = Graph[e, GraphLayout → {"LayeredEmbedding", "RootVertex" → root},
    граф        укладка графа
  VertexLabels → Placed["Name", Center], VertexSize → 0.4,
  метки для вершин  расположен      центр      размер вершины
  VertexLabelStyle → Directive[Italic, 28], EdgeShapeFunction →
  стиль меток вершин      директива   курсив          функция формы ребра
   GraphElementData["FilledArrow", "ArrowSize" → 0.05], EdgeStyle → Thick]
                                                        стиль ребра   жирный
```

Out[3]=



Лабораторная 4.

In[4]:=
```
Is = 10;
ug = UndirectedGraph[g];
     ненаправленный граф
pred = ConstantArray[0, Is];(*список предков*)
       постоянный массив
depth = ConstantArray[0, Is]; (*список глубин узлов*)
        постоянный массив
dinast = ConstantArray[0, Is];(*династичесий обход*)
         постоянный массив
posl = {};                    (*последовательность обхода*)
DepthFirstScan[ug, root, {"FrontierEdge" → Function[edge, {
                                           функция
       pred[[edge[[2]]]] = edge[[1]],
       depth[[edge[[2]]]] = depth[[edge[[1]]]] + 1
      }],
   "PrevisitVertex" → Function[vertex,
                      функция
      {If[posl ≠ {}, dinast[[posl[[-1]]]] = vertex], AppendTo[posl, vertex]}]
       условный оператор                              добавить в конец к
  }];
dinast[[posl[[-1]]]] = root;
(*Print[Range[Is]];
  печа… диапазон
Print[pred];
печатать
Print[depth];
печатать
Print[dir];
печатать
Print[dinast];
печатать
Print[posl];*)
печатать
```

In[12]:=
```
(*5. Списковые структуры вывести в виде таблицы
  (список связи или династического обхода можно не встраивать в таблицу,
    а вывести отдельным списком).*)
Grid[{Prepend[Range[Is], "i"], Prepend[pred, "pred[i]"], Prepend[depth, "depth[i]"],
табл… добавит… диапазон      добавить в начало          добавить в начало
  Prepend[dinast, "dinast[i]"]}, Frame → All, ItemStyle → {{Bold}, {Bold}}]
   добавить в начало          рамка  всё  стиль элемента  жирны…  жирный шрифт
Print[
печатать
 posl]
```

Out[12]=

i	1	2	3	4	5	6	7	8	9	10
pred[i]	0	1	1	1	3	3	4	5	6	6
depth[i]	0	1	1	1	2	2	2	3	3	3
dinast[i]	2	3	5	7	8	9	1	6	10	4

```
{1, 2, 3, 5, 8, 6, 9, 10, 4, 7}
```

Лабораторная 4. Дополнительное задание.

In[19]:=
```
(* Определение единственной цепи в дереве. *)
f1[x1_] := Block[{u1},
           программный блок
   u1 = NestWhileList[pred[[#]] &, x1, pred[[#]] ≠ 0 &]
        список итераций до
  ];
f1[10]
f1[7]
f1[5]

(*While[pred[[i]]≠0,Print[pred[[i]]];
  цикл–пока          печатать
 i=pred[[i]]]*)
```

Out[20]= {10, 6, 3, 1}

Out[21]= {7, 4, 1}

Out[22]= {5, 3, 1}

In[23]:=
```
(* Определение длины пути между двумя любыми вершинами дерева (плюс сам путь).*)
f2[x2_, y2_] := Block[{ux, uy, h1, h2, x, y},
                программный блок
   x = x2;
   y = y2;
   h1 = depth[[x2]];
   h2 = depth[[y2]];
   ux = {x};
   uy = {y};
   While[h1 ≠ h2, If[h1 > h2, {x = pred[[x]], h1 = h1 – 1, AppendTo[ux, x]},
   цикл–пока      условный оператор                       добавить в конец к
      {y = pred[[y]], h2 = h2 – 1, PrependTo[uy, y]}]];
                                   добавить в начало к
   While[x ≠ y, {x = pred[[x]], AppendTo[ux, x], y = pred[[y]], PrependTo[uy, y]}];
   цикл–пока                   добавить в конец к                добавить в начало к
   ux = Flatten[Append[ux, Drop[uy, 1]]];
        уплостить добавить в ⋯ отбросить
   List[Length[ux] – 1, ux]
   спи⋯ длина
  ];
f2[9, 5]
f2[3, 10]
f2[10, 10]
```

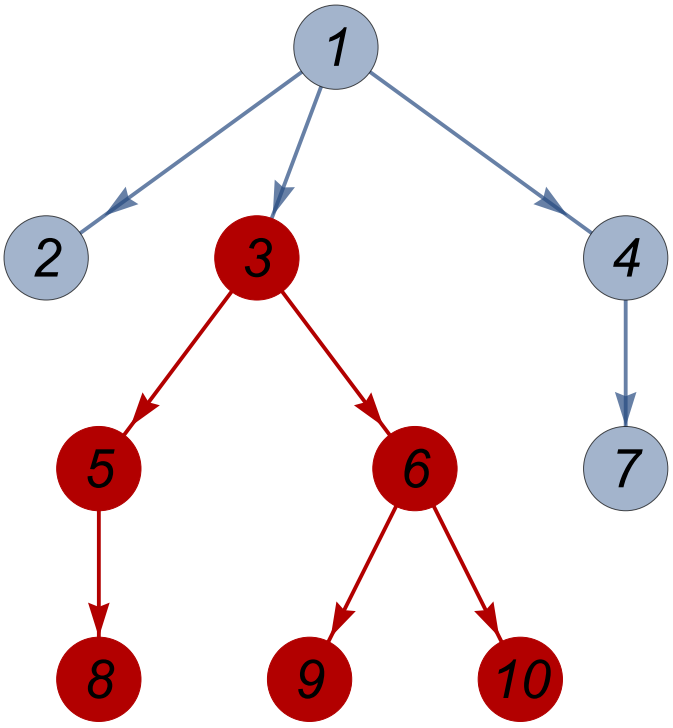
Out[24]= {3, {9, 6, 3, 5}}

Out[25]= {2, {3, 6, 10}}

Out[26]= {0, {10}}

In[27]:=
```
(* Нахождение поддерева с корнем в заданном узле.*)
f3[k_] := Block[{d, i, n, u},
          программный блок
   d = depth[[k]];
   i = dinast[[k]];
   n = 1;
   u = NestWhileList[dinast[[#]] &, i, depth[[#]] > d &];
       список итераций до
   PrependTo[u, k];
   добавить в начало к
   Drop[u, -1]
   отбросить
  ];
HighlightGraph[g, Subgraph[g, f3[3]]]
граф с подкраской   подграф
f3[1]
f3[8]
```

Out[28]=



Out[29]= {1, 2, 3, 5, 8, 6, 9, 10, 4, 7}

Out[30]= {8}

```
(* Определение всех листьев дерева (первый вариант).*)
f4[r_] := Block[{i, l},
          программный блок
   i = dinast[[r]];
   l = {};
   While[i ≠ r,
   цикл–пока
    {If[depth[[i]] ≥ depth[[dinast[[i]]]], AppendTo[l, i]], i = dinast[[i]]}];
     условный оператор                    добавить в конец к
   Return[l]
   вернуть управление
  ];
f4[root]
```

Out[15]= {2, 8, 9, 10, 7}

In[17]:=
```
(* Определение всех листьев дерева (второй вариант).*)
f5[r_] := Block[{i, l},
          программный блок
   i = dinast[[r]];
   l = {};
   While[i ≠ r, {If[i ≠ pred[[dinast[[i]]]], AppendTo[l, i]], i = dinast[[i]]}];
   цикл–пока      условный оператор                  добавить в конец к
   Return[l]
   вернуть управление
  ];
f5[root]
```

Out[18]= {2, 8, 9, 10, 7}