Вес поддерева – количество его концевых вершин. Баланс вершины – разница между весами правого и левого поддерева. Распечатать поддеревья с максимальным по модулю весом.

Myfun.cpp:

```
#include <cstdlib>
#include <limits.h>
#include "tree.h"

using namespace std;

void leaves_count(TreeNode *p, int *counter)
{
    if (p)
    {
        if (p->left == NULL && p->right == NULL) (*counter)++;
        else
        {
            leaves_count(p->left, counter);
            leaves_count(p->right, counter);
        }
    }
}

void T_balance(TreeNode *p, int *max_mod)
{
    if (p)
    {
        T_balance(p->left, max_mod);
        T_balance(p->right, max_mod);

        int counter_left = 0;
        int counter_right = 0;

        leaves_count(p->left, &counter_left);
        leaves_count(p->right, &counter_right);

        p->balance = counter_right - counter_left;

        if (abs(p->balance) > *max_mod) *max_mode = abs(p->balance);

    }
}

void T_lookup(TreeNode *p, int max_mod)
{
    if (p)
    {
        T_lookup(p->left, max_mod);
        T_lookup(p->right, max_mod);

        if (abs(p->balance) == max_mod)
        {
            T_Print(stdout,tr.root);
            T_Print(fout,tr.root);
        }
    }
}
```

Main.h:

```
#include "tree.h"

int  main(void)
{
    FILE *fin, *fout;

      int max_mod;

    if (!(fout = fopen("tree.res","w")))
        { printf("\n Cannot open file tree.res\n"); return -1; }
    if (!(fin = fopen("tree.dat","r")))
        { printf("\n Cannot open file tree.dat\n"); return -1; }

    Tree tr(fin);
    fclose(fin);

    printf("\nThe tree is :\n");
    T_Print(stdout,tr.root);
    T_Print(fout,tr.root);

      max_mod = 0;
      T_balance(tr.root, &max_mod);

    printf("\nThe tree with balances is :\n");
    T_Print(stdout,tr.root);
    T_Print(fout,tr.root);

      T_lookup(tr.root, max_mod);


    fclose(fout);
    return 0;
}
```

2.

Назовём весом поддерева наибольшую длину его ветвей (считая от его корня до концевых вершин, не имеющих обоих потомков), а балансом поддерева -- разность между весами его правого и левого поддеревьев. Требуется проставить в данном дереве правильные балансы в каждой его вершине, распечатать полученное дерево, а также определить, каких вершин больше в дереве -- с положительным балансом, с нулевым или с отрицательным?

Myfun.cpp:

```
#include <cstdlib>
#include <limits.h>
#include "tree.h"

using namespace std;

void maxlen(TreeNode *p, int level, int *longest)
{
      if (p)
      {
            if (p->left == NULL && p->right == NULL)
            {
```

```
                if (*longest < level) *longest = level;
            }
            else
            {
                maxlen(p->left, level + 1, longest);
                maxlen(p->right, level + 1, longest);
            }
      }
}

void T_balance(TreeNode *p, int *counter)
{
      if (p)
      {
            T_balance(p->left, counter);
            T_balance(p->right, counter);

            int longest_left = 0;
            int longest_right = 0;

            maxlen(p->left, 1, &longest_left);
            maxlen(p->right, 1, &longest_right);

            p->balance = longest_right - longest_left;

            if (p->balance > 0) counter[0]++;
            else if (p->balance == 0) counter[1]++;
            else counter[2]++;
      }
}
```

Main.h:

```
#include "tree.h"
#include <limits.h>

int  main(void)
{
      FILE *fin, *fout;

      if (!(fout = fopen("tree.res","w")))
            { printf("\n Cannot open file tree.res\n"); return -1; }
      if (!(fin = fopen("tree.dat","r")))
            { printf("\n Cannot open file tree.dat\n"); return -1; }

      Tree tr(fin);
      fclose(fin);

      printf("\nThe tree is :\n");
      T_Print(stdout,tr.root);
      T_Print(fout,tr.root);

      int counter[3] = {0,0,0};

      T_balance(tr.root, counter);
      printf("\nThe tree with balances is :\n");
      T_Print(stdout,tr.root);
      T_Print(fout,tr.root);
```

```
	printf("balance counters (+ 0 -): %i %i %i\n", counter[0], counter[1],
counter[2]);

	if (counter[0] > counter[1] && counter[0] > counter[2])
	{
		printf("positive balances prevail: %i\n", counter[0]);
	}
	else if (counter[1] > counter[0] && counter[1] > counter[2])
	{
		printf("zero balances prevail: %i\n", counter[1]);
	}
	else if (counter[2] > counter[0] && counter[2] > counter[1])
	{
		printf("negative balances prevail: %i\n", counter[2]);
	}

	fclose(fout);
	return 0;
}
```

3.

Назовём балансом поддерева число, равное разности между длинами самой длинной и самой короткой его ветвей (длина ветви -- количество вершин в цепочке от корня до вершины, не имеющей соответствующего потомка). Требуется проставить во всех вершинах балансы соответствующих поддеревьев и распечатать полученное дерево. После этого требуется определить какие значения балансов встречаются в дереве более одного раза и распечатать только различные эти значения в порядке возрастания (т.е., например, если балансы в дереве есть 1,5,1,3,2,2, то печатается 1,2).

Myfun.cpp:

```
#include <cstdlib>
#include <limits.h>
#include "tree.h"

using namespace std;

void maxlen(TreeNode *p, int level, int *longest)
{
	if (p)
	{
```

```
            if (p->left == NULL && p->right == NULL)
            {
                  if (*longest < level) *longest = level;
            }
            else
            {
                  maxlen(p->left, level + 1, longest);
                  maxlen(p->right, level + 1, longest);
            }
      }
}

void minlen(TreeNode *p, int level, int *shortest)
{
      if (p)
      {
            if (p->left == NULL || p->right == NULL)
            {
                  if (*shortest > level) *shortest = level;
            }
            else
            {
                  minlen(p->left, level + 1, shortest);
                  minlen(p->right, level + 1, shortest);
            }
      }
}

void T_balance(TreeNode *p)
{
      if (p)
      {
            T_balance(p->left);
            T_balance(p->right);

            int longest = INT_MIN;
            int shortest = INT_MAX;

            maxlen(p, 0, &longest);
            minlen(p, 0, &shortest);

            p->balance = longest - shortest;
      }
}

void T_lookup(TreeNode *p, int balance, int *count)
{
      if (p)
      {
            T_lookup(p->left, balance, count);
            T_lookup(p->right, balance, count);

            if (p->balance == balance) (*count)++;
      }
}
```

Main.h:

```
#include "tree.h"
```

```
#include <limits.h>

int  main(void)
{
    FILE *fin, *fout;

    if (!(fout = fopen("tree.res","w")))
        { printf("\n Cannot open file tree.res\n"); return -1; }
    if (!(fin = fopen("tree.dat","r")))
        { printf("\n Cannot open file tree.dat\n"); return -1; }

    Tree tr(fin);
    fclose(fin);

    printf("\nThe tree is :\n");
    T_Print(stdout,tr.root);
    T_Print(fout,tr.root);

      T_balance(tr.root);
    printf("\nThe tree with balances is :\n");
    T_Print(stdout,tr.root);
    T_Print(fout,tr.root);

    printf("\nBalances : ");
      for (int i = 0; i <= tr.root->balance; i++)
      {
            int j = 0;
            T_lookup(tr.root, i, &j);
            if (j > 1) printf(" %i", i);
      }
    printf("\n");

    fclose(fout);
    return 0;
}
```

8.

Назовём весом поддерева количество его вершин (включая корень), а балансом поддерева -- разность между весами его правого и левого поддеревьев. Требуется проставить в данном дереве правильные балансы в каждой его вершине, распечатать полученное дерево, а также определить, каких вершин больше в дереве -- с положительным балансом, с нулевым или с отрицательным?

Myfun.cpp:

```
#include <cstdlib>
#include <limits.h>
#include "tree.h"

using namespace std;

void nodes(TreeNode *p, int *n)
{
      if (p)
      {
            nodes(p->left, n);
            nodes(p->right, n);
```

```
            (*n)++;
      }
}

void T_balance(TreeNode *p, int *counter)
{
      if (p)
      {
            T_balance(p->left, counter);
            T_balance(p->right, counter);

            int weight_left = 0;
            int weight_right = 0;

            nodes(p->left, &weight_left);
            nodes(p->right, &weight_right);

            p->balance = weight_right - weight_left;

            if (p->balance > 0) counter[0]++;
            else if (p->balance == 0) counter[1]++;
            else counter[2]++;
      }
}
```

Main.h:

```
#include "tree.h"
#include <limits.h>

int  main(void)
{
      FILE *fin, *fout;

      if (!(fout = fopen("tree.res","w")))
            { printf("\n Cannot open file tree.res\n"); return -1; }
      if (!(fin = fopen("tree.dat","r")))
            { printf("\n Cannot open file tree.dat\n"); return -1; }

      Tree tr(fin);
      fclose(fin);

      printf("\nThe tree is :\n");
      T_Print(stdout,tr.root);
      T_Print(fout,tr.root);

      int counter[3] = {0,0,0};

      T_balance(tr.root, counter);
      printf("\nThe tree with balances is :\n");
      T_Print(stdout,tr.root);
      T_Print(fout,tr.root);

      printf("balance counters (+ 0 -): %i %i %i\n", counter[0], counter[1],
counter[2]);

      if (counter[0] > counter[1] && counter[0] > counter[2])
      {
```

```
            printf("positive balances prevail: %i\n", counter[0]);
      }
      else if (counter[1] > counter[0] && counter[1] > counter[2])
      {
            printf("zero balances prevail: %i\n", counter[1]);
      }
      else if (counter[2] > counter[0] && counter[2] > counter[1])
      {
            printf("negative balances prevail: %i\n", counter[2]);
      }

      fclose(fout);
      return 0;
}
```

13.

Назовём весом поддерева максимальное значение value среди всех его вершин, включая корень (для пустого дерева вес равен 0), а балансом поддерева -- разность между весами его правого и левого поддеревьев. Требуется проставить в данном дереве правильные балансы в каждой его вершине, распечатать полученное дерево, а также определить, каких вершин больше в дереве -- с положительным балансом, с нулевым или с отрицательным?

Myfun.cpp:

```
#include <cstdlib>
#include <limits.h>
#include "tree.h"

using namespace std;

void maxval(TreeNode *p, int *n)
{
      if (p)
      {
            maxval(p->left, n);
            maxval(p->right, n);
```

```
            if (*n < p->value)
            {
                  *n = p->value;
            }
      }
}

void T_balance(TreeNode *p, int *counter)
{
      if (p)
      {
            T_balance(p->left, counter);
            T_balance(p->right, counter);

            int weight_left = INT_MIN;
            int weight_right = INT_MIN;

            if (p->left) maxval(p->left, &weight_left); else weight_left = 0;
            if (p->right) maxval(p->right, &weight_right); else weight_right = 0;

            p->balance = weight_right - weight_left;

            if (p->balance > 0) counter[0]++;
            else if (p->balance == 0) counter[1]++;
            else counter[2]++;
      }
}
```

Main.h:

```
#include "tree.h"
#include <limits.h>

int  main(void)
{
      FILE *fin, *fout;

      if (!(fout = fopen("tree.res","w")))
            { printf("\n Cannot open file tree.res\n"); return -1; }
      if (!(fin = fopen("tree.dat","r")))
            { printf("\n Cannot open file tree.dat\n"); return -1; }

      Tree tr(fin);
      fclose(fin);

      printf("\nThe tree is :\n");
      T_Print(stdout,tr.root);
      T_Print(fout,tr.root);

      int counter[3] = {0,0,0};

      T_balance(tr.root, counter);
      printf("\nThe tree with balances is :\n");
      T_Print(stdout,tr.root);
      T_Print(fout,tr.root);

      printf("balance counters (+ 0 -): %i %i %i\n", counter[0], counter[1],
counter[2]);
```

```
	if (counter[0] > counter[1] && counter[0] > counter[2])
	{
		printf("positive balances prevail: %i\n", counter[0]);
	}
	else if (counter[1] > counter[0] && counter[1] > counter[2])
	{
		printf("zero balances prevail: %i\n", counter[1]);
	}
	else if (counter[2] > counter[0] && counter[2] > counter[1])
	{
		printf("negative balances prevail: %i\n", counter[2]);
	}

	fclose(fout);
	return 0;
}
```