Олимпиада школьников «Шаг в будущее» по общеобразовательному предмету "Информатика". 2014 год. Отборочный этап. 8-9 классы. Билет 1.

Задача 1: Ряд Фибоначчи (10)

Последовательность чисел 1, 1, 2, 3, 5, 8, 13, 21, …, a_n, …, два первых члена которой равны 1, а каждый член, начиная с третьего, равен сумме двух предыдущих: $a_{n+2} = a_{n+1} + a_n$, называется «рядом» Фибоначчи. Вычислить $a_n^2 - a_{n-1}a_{n+1}$.

Входные данные. Во входном файле записано одно целое число n ($3 \le n \le 10^{18}$).
Выходные данные. В выходной файл вывести одно целое число – значение вычисленного выражения.

Пример входного файла	Пример выходного файла
9	1

РЕШЕНИЕ:

```
#include "stdafx.h"

FILE *ifs, *ofs;

int _tmain(int argc, _TCHAR* argv[])
{
	unsigned long long n;
	ifs = fopen( "1.IN", "r" );
	fscanf( ifs, "%lld", &n );
	fclose( ifs );
	ofs = fopen( "1.OUT", "w" );
	if (n % 2 == 0)   // n - четное число
		fprintf( ofs, "%d\n", -1 );
	else        // n - нечетное число
		fprintf( ofs, "%d\n", 1 );
	fclose( ofs );
	return 0;
}
```

Задача 2: Сумма чисел (15)

Разобьем ряд натуральных чисел в группы: 1, (2, 3), (4, 5, 6), (7, 8, 9, 10), … . Найти сумму чисел n-й группы.

Входные данные. Во входном файле записано одно целое число n – номер группы ($1 \le n \le 10^6$).
Выходные данные. В выходной файл вывести одно целое число – значение вычисленной суммы.

Примеры входного файла	Примеры выходного файла
10	505
100	500050

РЕШЕНИЕ:

```
#include "stdafx.h"

FILE *ifs, *ofs;

int _tmain(int argc, _TCHAR* argv[])
{
	unsigned long long n;
	ifs = fopen( "2.IN", "r" );
	fscanf( ifs, "%lld", &n );
	fclose( ifs );
	ofs = fopen( "2.OUT", "w" );
// Аналитическое решение
	fprintf( ofs, "%lld\n", n*(n*n+1)/2 );
// Алгоритмическое решение
	unsigned long long m, s, i, j;
	m = n*(n-1)/2;
	s = 0;
	for (i=1, j=m+1; i<n+1; i++, j++) s += j;
```

```
	fprintf( ofs, "%lld\n", s );
	fclose( ofs );
	return 0;
}
```

Задача 3: Текст (20)

Местоимениями в английском языке будем считать следующие слова: 'I', 'he', 'she', 'it', 'me', 'him', 'her', 'his', 'we', 'you', 'they', 'us', 'them'. Дан текст на английском языке. Предложения в тексте начинаются с заглавной буквы, а в конце предложения ставится '.' (точка). Слова в предложениях и сами предложения отделяются друг от друга, по крайней мере, одним пробелом. Между словами допускаются символы пунктуации: ',' (запятая), ';' (точка с запятой), ':' (двоеточие), '–' (тире) и символ апострофа. Найти и напечатать местоимения, обнаруженные в тексте, и их количество.

Входные данные. Во входном файле записан текст на английском языке. Длина строки текста не более 60. Количество строк текста не более 1000. Длина слова не более 20.
Выходные данные. Местоимения должны быть напечатаны в верхнем регистре и в том порядке, в котором они впервые встречаются в исходном тексте. Каждое местоимение и его количество печатать с новой строки. Если местоимений в тексте нет, напечатать 'NO'.

Пример входного файла	Пример выходного файла
I'll tell you a story, a story anon, Of a noble prince, and his name was King John. For he was a prince, and a prince of great might, He held up great wrongs, he put down great right.	I 1 YOU 1 HIS 1 HE 3

РЕШЕНИЕ:

```
#include "stdafx.h"
#include <string.h>

FILE *ifs, *ofs;
const char *pronouns[13] = { "I", "HE", "SHE", "IT", "ME", "HIM", "HER", "HIS", "WE",
"YOU", "THEY", "US", "THEM" };
int ispronouns( const char *word);

int _tmain(int argc, _TCHAR* argv[])
{
	char line[100], *copy, *token;
	char seps[] = " .,;:-'\n\r";
	int counts[13] =  { 0 };
	int i, index;
	ifs = fopen( "3.IN", "r");
	ofs = fopen( "3.OUT", "w");
	while (!feof( ifs ))
	{
		if (fgets( line, 100, ifs ) != NULL)
		{
			copy = strupr( strdup( line ) );
			token = strtok( copy, seps );
			while (token != NULL)
			{
				/*fprintf( ofs, "%s\n", token );*/
				index = ispronouns( token );
				if (index != (-1)) counts[index]++;
				token = strtok( NULL, seps );
			}
		}
	}
	for (i = 0; i < 13; i++)
		if (counts[i] != 0)
			fprintf( ofs, "%s\t%d\n", pronouns[i], counts[i] );
	fclose( ifs );
	fclose( ofs );
	return 0;
}

int ispronouns( const char *word)
```

```
{
	int i, result;
	for (i = 0; i < 13; i++)
	{
		result = strcmp( word, pronouns[i] );
		if (result == 0) return i;
	}
	return -1;
}
```

Задача 4: Выражение (25)

Обычный метод записи математических выражений, в которых бинарный оператор записывается между операндами, известен под названием инфиксной записи. При отсутствии скобок операции выполняются согласно правилам приоритета операторов. Для изменения порядка выполнения операций применяют скобки. Префиксная польская запись – это форма записи математических выражений, в которой оператор располагается слева от операндов. Если оператор имеет фиксированную арность, то в такой записи будут отсутствовать скобки, и она может быть интерпретирована без неоднозначности. Для заданного инфиксного арифметического выражения получить последовательность операторов в префиксной польской записи.

Входные данные. Входной файл содержит одну строку, в которой записано инфиксное арифметическое выражение. В исходном выражении нет пробелов, в качестве операндов используются латинские буквы в верхнем или нижнем регистре, в качестве операторов используются знаки "+", "–", "*" и "/". Длина исходного выражения не превосходит 100.
Выходные данные. В выходной файл вывести одну строку, в которой записаны операторы префиксной польской записи.

Пример входного файла	Пример выходного файла
(a-b/c)/(d*e-(f+g*h))	/-/-*+*

РЕШЕНИЕ:

```
#include "stdafx.h"
#include <ctype.h>
#include <string.h>

const int N = 100;

typedef struct btree BTREE;

struct btree {
	char info;
	BTREE *left;
	BTREE *right;
};

FILE *ifs, *ofs;

static int op_stack[N+1];
static int ex_stack[N+1];
static int op_index = 0, ex_index = 0;
BTREE *root = 0;

void op_push( int value )
{
	op_stack[++op_index] = value;
}

int op_pop()
{
	op_index--;
	return (op_stack[op_index+1]);
}

int op_top()
{
	return (op_stack[op_index]);
}
```

```
void ex_push( int value )
{
      ex_stack[++ex_index] = value;
}

int ex_pop()
{
      ex_index--;
      return (ex_stack[ex_index+1]);
}

int ex_top()
{
      return (ex_stack[ex_index]);
}

int is_operator( int c )
{
      if (c == '+' || c == '-' || c == '/' || c == '*' || c == '^') return 1;
      else return 0;
}

int op_priority( int op )
{
      if (op == '^') return 4;
      if (op == '*' || op == '/') return 3;
      if (op == '+' || op == '-') return 2;
      return 1;
}

void printtree( BTREE *root )
{
      if (!isalpha( root->info )) fprintf( ofs, "%c", root->info ); // Печатаем текущий
узел
      if (root->left) printtree( root->left ); // Обходим левое поддерево
      if (root->right) printtree( root->right ); // Обходим правое поддерево
}

void buildtree( BTREE *&root )
{
      char top;
      if (ex_index == 0) return;
      top = ex_top();
      if (root == 0)
      {
            root = new BTREE;
            root->info = top;
            root->left = 0;
            root->right = 0;
      }
      if (is_operator( top ))
      {
            ex_pop();
            buildtree( root->right );
            buildtree( root->left );
      }
      else ex_pop();
}

int _tmain(int argc, _TCHAR* argv[])
{
      char line[N+1];
      int m, i;
      char ch, top;
      // Читаем строку выражения
      ifs = fopen( "4.IN", "r" );
      fgets( line, N, ifs );
```

```
	fclose( ifs );
	m = strlen( line );
	if (m == 0) return 1;
	// Реализуем алгоритм сортировочной станции
	op_stack[0] = 0;
	ex_stack[0] = 0;
	for (i = 0; i < m; i++)
	{
		ch = line[i];
		if (isalpha( ch )) ex_push( ch );
		else if (ch == '(') op_push( ch );
		else if (ch == ')')
		{
			top = op_top();
			while (top != '(' && op_index != 0)
			{
				ex_push( top );
				op_pop();
				top = op_top();
			}
			if (top == '(') op_pop();
		}
		else if (is_operator( ch ))
		{
			top = op_top();
			while (is_operator( top ) && op_priority( top ) >= op_priority( ch )
&& op_index != 0)
			{
				ex_push( top );
				op_pop();
				top = op_top();
			}
			op_push( ch );
		}
		else;
	}
	top = op_top();
	while (is_operator( top ) && op_index != 0)
	{
		ex_push( top );
		op_pop();
		top = op_top();
	}
	buildtree( root ); // Строим дерево
	ofs = fopen( "4.OUT", "w" );
	printtree( root ); // Обходим дерево
	fprintf( ofs, "\n" );
	fclose( ofs );
	return 0;
}
```

Задача 5: Площадь прямоугольников (30)

Ортогональную целочисленную решетку, состоящую из точек с целыми координатами в декартовой системе координат, будем обозначать через Z^2. На решетке Z^2 заданы N прямоугольников, стороны которых параллельны осям координат. Прямоугольники могут перекрываться. Каждый прямоугольник задается координатами левого нижнего угла (x_1, y_1) и координатами правого верхнего угла (x_2, y_2). Найти площадь, покрываемую этими прямоугольниками.

Входные данные. Первая строка входного файла содержит целое число N – количество прямоугольников ($1 \le N \le 1000$). В последующих N строках записаны четверки целых чисел x_1, y_1, x_2, y_2 ($0 \le x_1, y_1, x_2, y_2 \le 10^6$, $x_1 < x_2$ и $y_1 < y_2$), задающих координаты углов прямоугольников.
Выходные данные. В выходной файл вывести одно целое число – площадь, покрываемую прямоугольниками.

Примеры входного файла	Примеры выходного файла
3 0 6 20 16 14 0 24 10	376

50 50 60 60	
2 0 0 20 10 10 4 14 8	200

РЕШЕНИЕ:

```
#include "stdafx.h"
#include <algorithm>

using namespace std;

const int N = 1000;

int X1[N], Y1[N], X2[N], Y2[N];
int Y[2*N], L[N], R[N];
FILE *ifs, *ofs;

int _tmain(int argc, _TCHAR* argv[])
{
	int n, i, k, m, j, leftbound = 0;
	long long A;
	ifs = fopen( "5.IN", "r" );
	// Читаем количество прямоугольников
	fscanf( ifs, "%d\n", &n);
	// Читаем координаты углов прямоугольников
	for (i = 0; i < n; i++)
		fscanf( ifs, "%d %d %d %d\n", &X1[i], &Y1[i], &X2[i], &Y2[i] );
	fclose( ifs );
	// Собираем все Y-координаты и сортируем их
	for (i = 0; i < n; i++)
	{
		Y[2*i] = Y1[i];
		Y[2*i+1] = Y2[i];
	}
	sort(Y, Y+2*n);
	A = 0;
	// Цикл по всем блокам горизонтальных линий между двумя последовательными Y-
координатами
	for (k = 0; k < 2*n-1; k++)
	{
		// Определяем все прямоугольники в диапазоне
		m = 0;
		for (i = 0; i < n; i++)
			if (Y[k] >= Y1[i] && Y[k] < Y2[i])
			{
				L[m] = X1[i]; // Left side of photo
				R[m] = X2[i]; // Right side of photo
				m++;
			}
		// Сортируем все координаты
		sort(L, L+m);
		sort(R, R+m);
		// Сканируем слева направо
		for (i = j = 0; j < m;)
		{
			if (i < m && L[i] < R[j]) // Следующая координата - левая сторона
нового прямоугольника
			{
				if (i == j)
					leftbound = L[i]; // Устанавливаем новую границу
				i++;
			}
			else // Следующая координата - правая сторона старого прямоугольника
			{
				j++;
				if (i == j) // Добавляем к площади
```

```
                                A += (long long)(Y[k+1] - Y[k]) * (R[j-1] - leftbound);
                        }
                }
        }
        // Печатаем ответ
        ofs = fopen( "5.OUT", "w" );
        fprintf( ofs, "%lld\n", A);
        fclose( ofs );
        return 0;
}
```