Задание 1
Самый тяжелый гриб

Решение задачи состоит в нахождении условного максимума.

Вариант решения на языке C++

```
#include <iostream>
using namespace std;

int main() {
	long k, n;
	cin >> k >> n;
	long max = 0;
	for (long i = 0; i < n; ++i) {
		long a;
		cin >> a;
		if (a <= k && a > max) max = a;
	}
	cout << max;
	return 0;
}
```

Задание 2
Сколько влезет грибов

Для решения задачи необходимо задать массив с индексами от *1* до *V*. Индекс соответствует объему, а значение соответствующего элемента массива – максимальному весу грибов для данного занятого объема.

Вариант решения на языке C++

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main() {
	long V, n;
	cin >> V >> n;
	vector<long long> basket;
	basket.resize(V + 1, 0);
	for (long i = 0; i < n; ++i) {
		long long v,m;
		cin >> v>>m;
		if (v <= V) {
			for (long j=V-v;j>=0;--j)
				basket[v+j] = max(basket[v+j], basket[j]+m);
		}
	}
	long long  answer = 0;
	for (long i = 1; i <= V ; ++i)
		answer = max(basket[i], answer);
	cout << answer;
	return 0;
}
```

Задание 3
Собрать цветы

Для решения задачи необходимо найти все простые пути из *A* в *B*.

Вариант решения на языке C++

```
#include <iostream>
#include <vector>
#include <set>
#include <algorithm>
using namespace std;

struct vertex {
	vector<int> neighbors;
	long long flowers,weight;
};

vector<vertex> graph;
int A, B;

long long Flowers(set<int> s, int v) {
	if (v == B) return graph[v].flowers;
	long long res = 0;
	s.insert(v);
	for (int i = 0; i < graph[v].neighbors.size(); ++i) {
		int p = graph[v].neighbors[i];
		if (s.find(p) == s.end()) {
			long long k = Flowers(s, p);
			if (k>0) res = max(res,k+ graph[p].flowers);
		}
	}
	return res;
}

int main() {
	int  n, m;
	cin >> n >> m;
	graph.resize(n);
	for (int i = 0; i < n; ++i) {
		long long v, m;
		cin >> graph[i].flowers;
	}
	cin >> A >> B;
	--A; --B;
	for (long i = 0; i < m; ++i) {
		long long p, r;
		cin >> p >> r;
		graph[p - 1].neighbors.push_back(r - 1);
		graph[r - 1].neighbors.push_back(p - 1);
	}
	set<int> s;

	cout << Flowers(s, A)<<endl;
	system("pause");
	return 0;
}
```

Задание 4
Красивый букет

При решении необходимо учитывать, что на каждой полянке собираются цветы одного или двух видов. Количество различных букетов – шесть. Первоначально собирается максимальное количество цветов в каждом букете. Количество цветов в полученном букете не кратно 5, то количество цветов уменьшается на минимально возможное значение.

Вариант решения на языке C++

```
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
	long long sum_r, sum_l, sum_v, sum_rl, sum_rv, sum_lv, m_r[5], m_l[5], m_v[5],
m_rl[5], m_rv[5], m_lv[5];
	sum_r = sum_l = sum_v = sum_rl = sum_rv = sum_lv = 0;
	for (int i = 0; i < 5; ++i)
		m_r[i] = m_l[i] = m_v[i] = m_rl[i] = m_rv[i] = m_lv[i] = 100001;
	long long n;
	cin >> n;

	for (long long i = 0; i < n; ++i) {
		long long a, b, c;
		cin >> a >> b>>c;
		sum_r += a;
		sum_l += b;
		sum_v += c;
		sum_rl += a + b;
		sum_rv += a + c;
		sum_lv += b + c;
		m_r[a % 5] = min(m_r[a % 5], a);
		for (int k= a % 5+1;k<5;++k)
		  m_r[k] = min(m_r[k], m_r[k-a % 5]+a);
		m_l[b % 5] = min(m_l[b % 5], b);
		for (int k= b % 5+1;k<5;++k)
		    m_l[k] = min(m_l[k], m_l[k-b % 5]+b);
		m_v[c % 5] = min(m_v[c % 5], c);
		for (int k= c % 5+1;k<5;++k)
		  m_v[k] = min(m_v[k], m_v[k-c % 5]+c);
		m_rl[(a + b) % 5] = min(m_rl[(a + b) % 5], a + b);
		m_rl[(b) % 5] = min(m_rl[(b) % 5], b);
		m_rl[(a) % 5] = min(m_rl[(a) % 5], a);
		for (int k= (a + b)  % 5+1;k<5;++k)
		  m_rl[k] = min(m_rl[k], m_rl[k-(a + b) % 5]+ (a + b));
		for (int k= a  % 5+1;k<5;++k)
		  m_rl[k] = min(m_rl[k], m_rl[k-a % 5]+ a);
		for (int k=b % 5+1;k<5;++k)
		  m_rl[k] = min(m_rl[k], m_rl[k-b% 5]+ b);
		m_rv[(a + c) % 5] = min(m_rv[(a + c) % 5], a + c);
		m_rv[a % 5] = min(m_rv[a  % 5], a );
		m_rv[c % 5] = min(m_rv[c % 5],  c);
		for (int k= (a + c)  % 5+1;k<5;++k)
		  m_rv[k] = min(m_rv[k], m_rv[k-(a + c) % 5]+ (a + c));
		for (int k= a  % 5+1;k<5;++k)
		  m_rv[k] = min(m_rv[k], m_rv[k-a % 5]+ a);
		for (int k=c % 5+1;k<5;++k)
		  m_rv[k] = min(m_rv[k], m_rv[k-c% 5]+ c);
		m_lv[(b + c) % 5] = min(m_lv[(b + c) % 5], b + c);
		m_lv[b % 5] = min(m_lv[b % 5], b );
		m_lv[c % 5] = min(m_lv[c % 5],c);
```

```
            for (int k= (c + b)  % 5+1;k<5;++k)
              m_lv[k] = min(m_lv[k], m_lv[k-(c + b) % 5]+ (c + b));
            for (int k= c  % 5+1;k<5;++k)
              m_lv[k] = min(m_lv[k], m_lv[k-c % 5]+ c;
            for (int k=b % 5+1;k<5;++k)
              m_lv[k] = min(m_lv[k], m_lv[k-b% 5]+ b);
        }
        m_r[0] = m_l[0] = m_v[0] = m_rl[0] = m_rv[0] = m_lv[0] = 0;
        if (sum_r % 5 != 0) {
            n = sum_r % 5;
            if (m_r[n] == 100001) sum_r = 0; else sum_r -= m_r[n];
        }
        if (sum_l % 5 != 0) {
            n = sum_l % 5;
            if (m_l[n] == 100001) sum_l = 0; else sum_l -= m_l[n];
        }
        if (sum_v % 5 != 0) {
            n = sum_v % 5;
            if (m_v[n] == 100001) sum_v = 0; else sum_v -= m_v[n];
        }
        if (sum_rl % 5 != 0) {
            n = sum_rl % 5;
            if (m_rl[n] == 100001) sum_rl = 0; else sum_rl -= m_rl[n];
        }
        if (sum_rv % 5 != 0) {
            n = sum_rv % 5;
            if (m_rv[n] == 100001) sum_rv = 0; else sum_rv -= m_rv[n];
        }
        if (sum_lv % 5 != 0) {
            n = sum_lv % 5;
            if (m_lv[n] == 100001) sum_lv = 0; else sum_lv -= m_lv[n];
        }
        long long  answer = max(sum_r, sum_l);
        answer = max(answer, sum_v);
        answer = max(answer, sum_rl);
        answer = max(answer, sum_rv);
        answer = max(answer, sum_lv);
        cout << answer;
        system("pause");
        return 0;
}
```

Задание 5
Воздушный мост

Введем числовую ось с началом координат, соответствующем домику Кубика. Тогда координата $(i+1)$-го дерева равна $x_{i+1}= a_1+a_2+\ldots+a_i$ ($x_1=0$). Рассмотрим множества отрезков $R=\{(a,b)|\ a=x_i,\ b=x_i+l_i,\ 1\le i<n\}$ и $L=\{(a,b)|\ a=x_i-l_i,\ b=x_i,\ 1<i\le n\}$. Задача имеет положительное решение если множество $R\cup L$ покрывает отрезок $[x_1, x_n]$.

Вариант решения на языке C++

```
#include <vector>
using namespace std;

int main() {
	int n;
	cin >> n;
	vector<int> a(n), l(n);
	a[0] = 0;
	for (int i = 1; i < n; ++i)
		cin >> a[i];

	for (int i = 0; i < n; ++i)
		cin >> l[i];
	int ltree=0, lfree=l[0];
	while (ltree<n - 1 && lfree>a[ltree + 1]) {
		++ltree;
		lfree = max(lfree - a[ltree], l[ltree]);
	}
	if (ltree == n - 2 && lfree+l[n-1] > a[n - 1]) {
		cout << "YES";
		return 0;
	}
	int rtree = n-1, rfree = l[n-1];
	while (rtree>0 && rfree>a[rtree]) {
		rfree = max(rfree - a[rtree], l[rtree-1]);
		--rtree;
	}
	if (rtree == 0) {
		cout << "YES";
		return 0;
	}

	if (rtree<ltree || (ltree+1==rtree)&&(lfree+ rfree>a[rtree])) cout << "YES"; else
cout << "NO";
	system("pause");
	return 0;
}
```

Задание 6
Построить ограду

Решение задачи состоит в нахождении периметра выпуклой оболочки заданного множества точек.

Вариант решения на языке C++

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
vector<int> x, y, v;
int n;

bool check_line(int a, int b) {
	int k;
	for (int i = 0; i < n; ++i) {
		if (((x[i] - x[a])*(y[b] - y[a]) - (y[i] - y[a])*(x[b] - x[a])) == 0) {
			if ((x[a] - x[i])*(x[a] - x[i]) + (y[a] - y[i])*(y[a] - y[i]) > (x[a]
- x[b])*(x[a] - x[b]) + (y[a] - y[b])*(y[a] - y[b])) return false;
		}
		else k = i;
	}
	for (int i = 0; i < n; i++) {
		if (i == k || i == a || i == b) continue;
		if (((x[i] - x[a])*(y[b] - y[a]) - (y[i] - y[a])*(x[b] - x[a]))*((x[k] -
x[a])*(y[b] - y[a]) - (y[k] - y[a])*(x[b] - x[a])) < 0) return false;
	}
	return true;
}

int next_point(int currrent) {
	for (int i = 0; i < n; ++i) {
		if (v[i] == 0 && check_line(currrent, i)) return i;
	}
	return -1;
}

int main() {

	cin >> n;
	x.resize(n);
	y.resize(n);
    v.resize(n,0);
	for (int i = 0; i < n; ++i)
		cin >> x[i]>>y[i];

	int start_point = 0;
	for (int i = 0; i < n; ++i) {
		if (x[i] < x[start_point] || x[i] == x[start_point] && y[i] <
y[start_point]) start_point = i;
	}
	v[start_point] = 1;
	int i, last= start_point;
	double length = 0;
	while ((i = next_point(last)) != -1) {
		v[i] = 1;
		length += sqrt((x[last] - x[i])*(x[last] - x[i]) + (y[last] -
y[i])*(y[last] - y[i]));
		last = i;
	}
```

```
	length += sqrt((x[last] - x[start_point])*(x[last] - x[start_point]) + (y[last] -
y[start_point])*(y[last] - y[start_point]));
	cout << length;
	system("pause");
	return 0;
}
```