Задание №1

Для решения задачи рекомендуется построить отсортированный массив. Перебирая значения от большего к меньшему произвести первого числа, который равен сумме четырех меньших его. Если такой элемент отсутствует, то выводиться шестое по величине значение.
Ниже приведен пример решения на языке C++.

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

void main()
{
    long long n;
    cin >> n;
    vector <long long> a;
    a.resize(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    sort(a.begin(), a.end());
    reverse(a.begin(), a.end());
    for (int i = 0; i < n; i++) {
        for (int i1 = i + 1; i1 < n; i1++) {
            for (int i2 = i1 + 1; i2 < n; i2++) {
                if (a[i] <= (a[i1] + a[i2]))  continue;
                for (int i3 = i2 + 1; i3 < n; i3++) {
                    if (a[i] <= (a[i1] + a[i2] + a[i3]))  continue;
                    for (int i4 = i3 + 1; i4 < n; i4++) {
                        if (a[i] == (a[i1] + a[i2] + a[i3] + a[i4])) {
                            cout << a[i];
                            return;
                        }
                        else if (a[i] < (a[i1] + a[i2] + a[i3] + a[i4])) break;
                    }
                }
            }
        }
    }
    cout << a[5];
}
```

Задание №2

Для решения задачи необходимо последовательно суммировать входные данные. Если очередное входное данное делает сумму больше вместимости кубуса, то счетчик линий посадки *k* (в один кубус помещается 2 линии) необходимо увеличить на 1, а сумму заново начать с последнего введенного числа. Ответ равен целой части от *(k+1)/2*.
Ниже приведен пример решения на языке C++.

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
void main() {
    long long n, m;
    cin >> n >> m;
    long long s = 0;
    long long k = 1;
    for (int i = 0; i < m; i++) {
        long long x;
        cin >> x;
        if (s + x > n) {
            s = 0;
            k = k + 1;
        }
        s += x;
    }
    cout << (k+1)/2;
}
```

Задание №3

Задача решается методами динамического программирования. Для решения задачи необходимо контролировать шесть пар значений *(грань, накопленная сумма)*. Если грань очередного кубика совпадает с *гранью*, то *накопленная сумма* увеличивается и ей в пару ставится грань, противоположная присоединенной.
Ниже приведен пример решения на языке C++.

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

typedef    long long  sides[6][2];
typedef   int cube[6];

int getSide(int i) {
    if (i % 2 == 1) return i - 1;
    return i + 1;
}


sides sum, tmp;

void check() {
    for (int i = 0; i < 6; i++)
        for (int j = i + 1; j < 6; j++)
            if (sum[i][0] == sum[j][0]) {
                sum[i][1] = max(sum[i][1], sum[j][1]);
                sum[j][1] = 0;
                sum[j][0] = 0;
            }
}

void main() {

    int n;

    cube a;
    cin >> n >> a[0] >> a[1] >> a[2] >> a[3] >> a[4] >> a[5];
    for (int i = 0; i < 6; i++) {
        sum[i][0] = a[i];
        sum[i][1] = 0;
    }

    for (int k = 1; k < n; k++) {
        check();
        cin >> a[0] >> a[1] >> a[2] >> a[3] >> a[4] >> a[5];
        for (int i = 0; i < 6; i++) {
            int i1 = getSide(i);
            tmp[i1][0] = 0;
            tmp[i1][1] = 0;
            for (int j = 0; j < 6; j++)
                if (a[i] == sum[j][0]) {
                    tmp[i1][0] = a[i1];
                    tmp[i1][1] = sum[j][1] + a[i];
                }
        }
        for (int i = 0; i < 6; i++) {
```

```
            sum[i][0] = tmp[i][0];
            sum[i][1] = tmp[i][1];
        }
    }
    long long m = 0;
    for (int i = 0; i < 6; i++)
        m = max(m, sum[i][1]);
    cout << 2 * m;
}
```

Задание №4

В силу большого набора данных, для решения рекомендуется предварительно сформировать массив для 10^7 чисел определяющего простоту чисел с помощью решета Эратосфена (например, $a_i = 1$, если i простое число, и $a_i = 0$ иначе).
Ниже приведен пример решения на языке C++.

```
#include <iostream>
#include <algorithm>

using namespace std;
const int sn = 10000001;

bool a[sn];
void Eratosfen() {
    for (int i = 0; i < sn; i++) a[i] = true;
    a[1] = false;
    for (int i = 2; i < sn; i++) {
        if (!a[i]) continue;
        for (int j = 2 * i; j < sn; j += i) a[j] = false;
    }
}

void main() {
    Eratosfen();
    long long t, k;
    cin >> t;
    k = 0;
    for (long long i = 0; i < t; ++i) {
        int cube[6];
        cin >> cube[0] >> cube[1] >> cube[2] >> cube[3] >> cube[4] >> cube[5];
        if (a[cube[1]] and a[cube[2]] and a[cube[3]] and
            a[cube[4]] and a[cube[5]] and a[cube[0]])  ++k;
    }
    cout << k;
}
```

Задание №5

При решении задачи целесообразно использовать системы непересекающихся множеств для определения компонент связанности. Разрушенные мосты необходимо отсортировать по стоимости и выбирать те, которые соединяют острова из разных компонент. При выборе моста соответствующие компоненты объединяются.

Ниже приведен пример решения на языке C++.

```
#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;

struct edge {
    int v1, v2, cost;
};

vector<int> parent;
vector<edge> broken;

void sortEdge() {
    for (int k = broken.size(); k > 0; k--)
        for (int i = 1; i < k; ++i)
            if (broken[i - 1].cost > broken[i].cost) {
                edge b = broken[i - 1];
                broken[i - 1] = broken[i];
                broken[i] = b;
            }
}

int DSU(int a) {
    if (parent[a] == a) return a;
    return parent[a] = DSU(parent[a]);
};

int main() {
    int n, m, d;
    cin >> n >> m >> d;
    parent.resize(n);
    broken.resize(d);
    for (int i = 0; i < parent.size(); ++i)
        parent[i] = i;
    for (int i = 0; i < m - d; ++i) {
        int v1, v2;
        cin >> v1 >> v2;
        int a = DSU(v1 - 1);
        int b = DSU(v2 - 1);
        if (a != b)parent[a] = b;
    }
    for (int i = 0; i < broken.size(); ++i) {
        int v1, v2, c;
        cin >> v1 >> v2 >> c;
        broken[i].v1 = v1 - 1;
        broken[i].v2 = v2 - 1;
        broken[i].cost = c;
    }
    sortEdge();
    long long answer = 0;
```

```
    for (int i = 0; i < broken.size(); ++i)
        if (DSU(broken[i].v1) != DSU(broken[i].v2)){
            parent[DSU(broken[i].v1)] = DSU(broken[i].v2);
            answer += broken[i].cost;
        }
    cout << answer;
    return 0;
}
```

Задание №6

Необходимо определить двух пар противоположных граней с заданными условием значением.

Ниже приведен пример решения на языке C++.

```
#include <iostream>
#include <algorithm>

using namespace std;
   bool is5(long long n) {
      long long sq = round(sqrt(n));
      if ((n == 1) or (sq * sq) != n)  return false;
      int k = 0;
      for (int i = 2; i < sq; ++i)
         if (n % i == 0) {
            k = k + 2;
            if (k > 2) return false;
         }
      return k == 2;
   }

   void main() {
      long long a1, a2, a3, a4, a5, a6;
      cin >> a1 >> a2 >> a3 >> a4 >> a5 >> a6;
      int k = 0;
      if (is5(a1) && is5(a2)) ++k;
      if (is5(a3) && is5(a4)) ++k;
      if ((k == 1) && is5(a5) && is5(a6)) ++k;
      if (k > 1) cout << "YES";
      else
      cout << "NO";
   }
```