

Pro uložení vstupních dat jsem použil pole, jehož index je velikost požadovaného kousku a jehož hodnota je počet požadovaných kusů této velikosti. Samotný algoritmus pak prochází toto pole od nejvyššího indexu (největších kousků) a ke každému kousku se snaží najít takový kousek, který by pizzu buď doplnil do celkové velikosti nebo ji alespoň zvětšil. Pokud je vytvořena celá pizza, nebo je zvětšena a není možné ji dále doplnit z dostupných kousků, je uložena do příslušného prvku pole a pokračuje se dalším kouskem stejné velikosti, pokud existuje. Pokud neexistuje, opakujeme stejný postup pro části pizzy o jeden kousek menší, až takto zpracujeme všechny.

Po provedení tohoto algoritmu nám v poli zůstanou části, které již nejde zvětšit (včetně celých pizz). Sečtením hodnot všech prvků pole tedy získáme počet celých pizz, které musí Marco upéct.

Čas provádění algoritmu bude růst s počtem objednaných kousků.

Množství použité paměti závisí na tom, na kolik kousků pizzu dělíme, je tedy konstantní.

```

#include <iostream>
#include <fstream>

//nulty index pole pouzijeme jako pocitadlo cellych pizz
#define WHOLE 0
//algoritmus je mozne pouzit pro pizzu o libovolnem poctu kousku
#define DIVISOR 6

using namespace std;

int main(void)
{
    ifstream infile("pizza.in");
    ofstream outfile("pizza.out");

    //pole, jehoz index je velikost kousku pizzy a hodnota pocet kousku
    //index 0 (WHOLE) - pocet cellych pizz
    int size[DIVISOR];
    for(int i=0; i<DIVISOR; i++) size[i] = 0;

    //nacteni poctu objednavek
    int numorders;
    if(!(infile >> numorders)){
        cerr << "Neplatny vstup\n";
        exit(EXIT_FAILURE);
    }

    //nacteni jednotlivych objednavek
    for(int i=0; i<numorders; i++){
        int pieces;
        if(!(infile >> pieces && pieces > 0)){
            cerr << "Neplatny vstup\n";
            exit(EXIT_FAILURE);
        }
        size[WHOLE] += pieces/DIVISOR;
        if(pieces%DIVISOR){
            size[pieces%DIVISOR]++;
        }
    }
}

```

```

//pro kazdy kousek pocinaje tim nejvetsim
for(int i=5; i>0; i--){
    while(size[i]){
        //ulozime si soucasnou velikost pizy
        int act_size=i;
        size[i]--;
        //pro kazdy mozny doplnek momentalniho kousku
        for(int s=(DIVISOR-i); s>0; s--){
            while(size[s]){
                //pokud muzeme kousek pripojit, udelame to
                if(act_size+s<=DIVISOR){
                    size[s]--;
                    act_size+=s;
                    //pokud ne, prejdeme na mensi kousek
                }else{
                    break;
                }
            }
            //pokud je pizza cela, nebudeme se ji snazit zvetsit
            if(act_size==DIVISOR) break;
        }
        //pizza je bud cela a nebo uz nejde udelat vetsi - peceme celou
        size[WHOLE]++;
    }
}

outfile << size[WHOLE] << '\n';

return EXIT_SUCCESS;
}

```