# 第一次算法设计作业

姓名： 学号：

【说明】共两道题，至少完成一题，全完成可加分。解答应包括解题思路（方法、时间复杂度分析）、代码（请在关键代码旁加上注释）以及按结果验证中的输入得到的输出截图，设计算法时需要考虑边界约束。作业提交时请只保留选做的题，删除空白题，并将程序代码一起打包发到助教邮箱。

【题一】 **分解质因数**

给定一个正整数num，返回分解成质因数的结果，要求应用递归实现。

【示例1】

输入：num =435234

输出： 2 3 17 17 251

【示例2】

输入：num =289

输出： 17 17

【示例3】

输入：num =78

输出： 2 3 13

【示例4】

输入：num =89

输出： 89

【解题思路】

如果当前数能被 divisor 整除，则 divisor 是一个质因数，继续分解 num / divisor。

如果不能整除，则尝试更大的 divisor（divisor + 1），直到 num 被分解为1。

递归终止条件为当 num 被分解到1时。

【代码】

#include <iostream>

#include <vector>

void primeFactors(int num, int divisor, std::vector<int>& result) {

if (num == 1) {

return;

}

if (num % divisor == 0) {

result.push\_back(divisor);

primeFactors(num / divisor, divisor, result);

}

else {

primeFactors(num, divisor + 1, result);

}

}

std::vector<int> decomposePrimeFactors(int num) {

std::vector<int> result;

primeFactors(num, 2, result);

return result;

}

int main() {

int num;

std::cin >> num;

if (num <= 1) {

std::cout << "请输入大于1的正整数！" << std::endl;

return 1;

}

std::vector<int> factors = decomposePrimeFactors(num);

for (int factor : factors) {

std::cout << factor << " ";

}

std::cout << std::endl;

return 0;

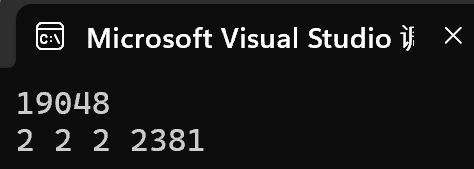
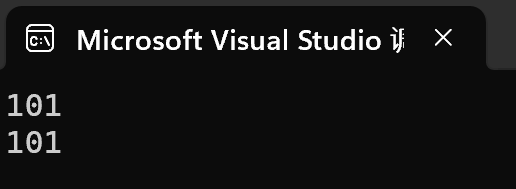
}

【结果验证】

1、输入：num =101

2、输入: num =654771

3、输入: num =19048



【题二】**最大子数组**   
给定一个整数数组nums，请用分治的方法找到有最大元素之和的子数组，并返回该子数组的元素之和（只返回元素之和，不用解释是哪个子数组）。

【示例1】输入：nums = [-2,1,-3,4,-1,2,1,-5,4]

输出：6

解释：子数组[4,-1,2,1]有最大元素之和6

【示例2】输入：nums = [1]

输出：1

解释：子数组[1]有最大元素之和1

【示例3】输入：nums = [5,4,-1,7,8]

输出：23

解释：子数组[5,4,-1,7,8]有最大元素之和23

【约束】

1 <= nums.length <= 105

-104 <= nums[i] <= 104

【解题思路】

输入的num数组不是直接的数字，而是一个带有[]的字符串。因此可以先借助getline函数读取用户输入的字符串，随后调用getNums函数解析字符串，将其转换为整数数组。随后送入接下来的函数中进行进一步运算。

将数组分为左半部分和右半部分两个部分，然后分别在这两部分中寻找最大子数组和。随后再将左半部分和右半部分再次分为左半部分和右半部分….以此类推直到无法再分。借助crossSum函数计算跨越中间的最大子数组和。借助helper函数将数组分为两部分，分别递归求解左半部分和右半部分的最大子数组和，最后返回这三个值中的最大值。

【代码】

#include <vector>

#include <algorithm>

#include <climits>

#include <iostream>

#include <sstream>

#include <string>

using namespace std;

// 计算跨越中间的最大和

int crossSum(vector<int>& v, int l, int m, int r) {

int lsum = INT\_MIN, sum = 0;

for (int i = m; i >= l; i--) {

sum += v[i];

if (sum > lsum) lsum = sum;

}

int rsum = INT\_MIN;

sum = 0;

for (int i = m + 1; i <= r; i++) {

sum += v[i];

if (sum > rsum) rsum = sum;

}

return lsum + rsum;

}

// 分治递归

int helper(vector<int>& v, int l, int r) {

if (l == r) return v[l];

int m = l + (r - l) / 2;

int left = helper(v, l, m);

int right = helper(v, m + 1, r);

int cross = crossSum(v, l, m, r);

return max({ left, right, cross });

}

// 主求解函数

int maxSubArr(vector<int>& v) {

return helper(v, 0, v.size() - 1);

}

// 输入解析

vector<int> getNums(string s) {

vector<int> res;

size\_t start = s.find('[');

size\_t end = s.find(']');

if (start == string::npos || end == string::npos) return res;

string arr = s.substr(start + 1, end - start - 1);

stringstream ss(arr);

string num;

while (getline(ss, num, ',')) {

res.push\_back(stoi(num));

}

return res;

}

int main() {

string s;

getline(cin, s);

vector<int> v = getNums(s);

if (v.empty()) {

cout << "输入格式错误！" << endl;

return 1;

}

cout << maxSubArr(v) << endl;

return 0;

}

【结果验证】1、输入：nums = [1,-5,3,-2,6,-1,4]

2、输入：nums = [-3,5,2,3,0,-1,8,-6,7,1]

