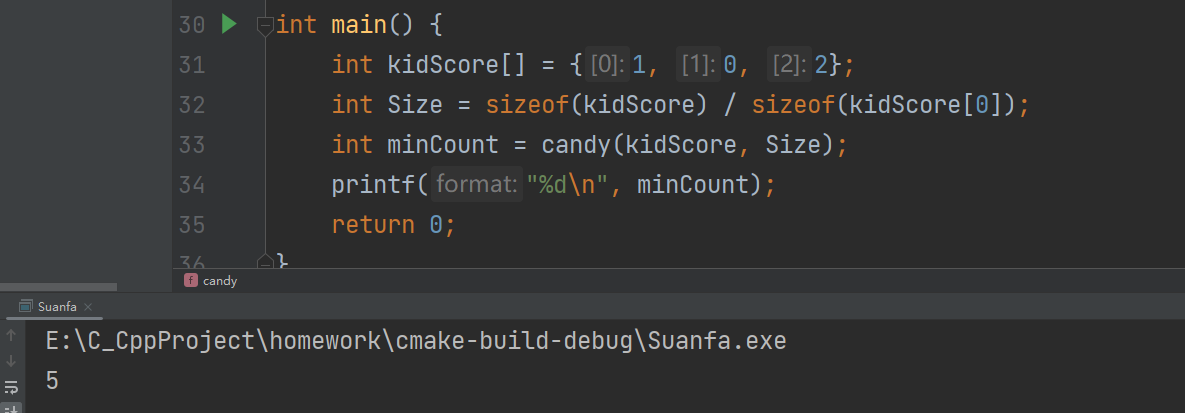
第一题

int maxProfit(int\* prices, int pricesSize, int fee) {  
 int buy = -prices[0];  
 int sell = 0;  
 for (int i = 1; i < pricesSize; i++) {  
 int temp = buy;  
 buy = (buy > sell - prices[i]) ? buy : (sell - prices[i]);  
 sell = (sell > temp + prices[i] - fee) ? sell : (temp + prices[i] - fee);  
 }  
 return sell;  
}



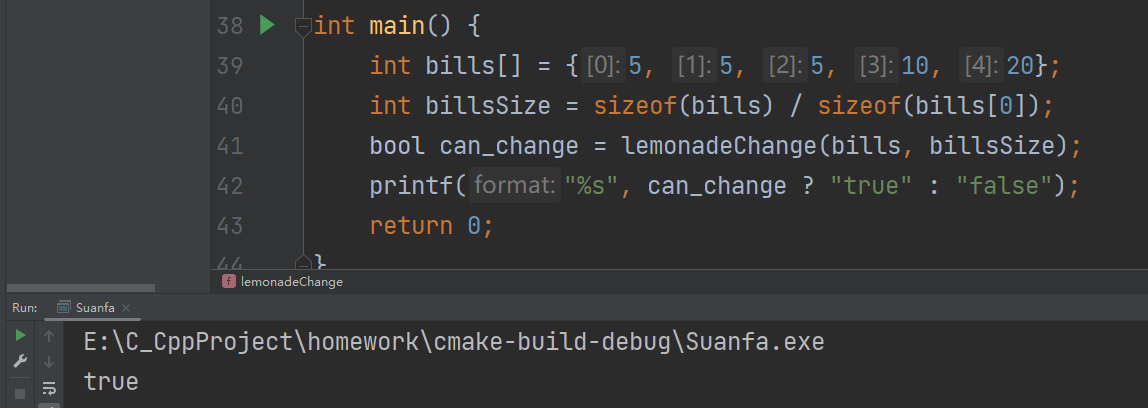
第二题

int candy(int \*kidScore, int Size) {  
 int \*candies = (int \*) malloc(Size \* sizeof(int));  
 for (int i = 0; i < Size; i++) {  
 candies[i] = 1;  
 }  
 for (int i = 1; i < Size; i++) {  
 if (kidScore[i] > kidScore[i - 1]) {  
 candies[i] = candies[i - 1] + 1;  
 }  
 }  
 for (int i = Size - 2; i >= 0; i--) {  
 if (kidScore[i] > kidScore[i + 1] && candies[i] <= candies[i + 1]) {  
 candies[i] = candies[i + 1] + 1;  
 }  
 }  
 int total = 0;  
 for (int i = 0; i < Size; i++) {  
 total += candies[i];  
 }  
 free(candies);  
 return total;  
}



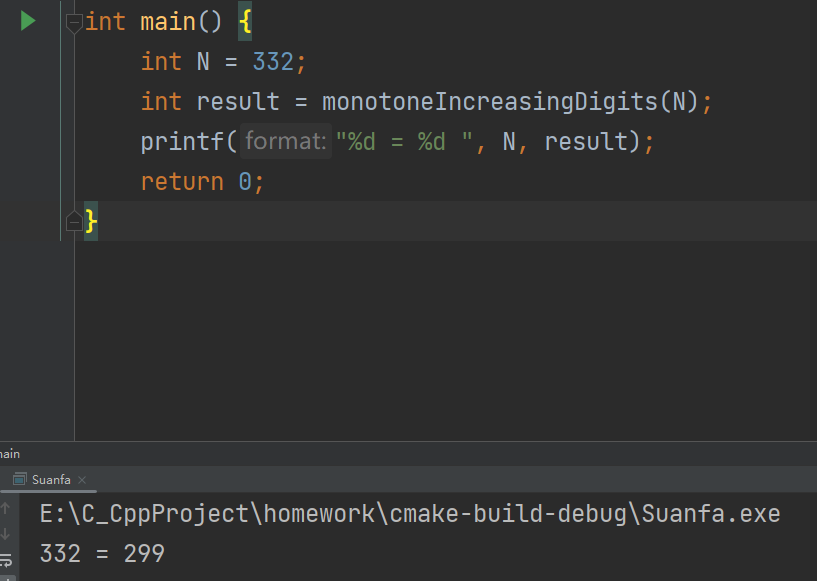
第三题

#define false 0  
#define true 1  
typedef int bool;  
bool lemonadeChange(int \*bills, int billsSize) {  
 int five = 0; // 5美元钞票的数量  
 int ten = 0; // 10美元钞票的数量  
  
 for (int i = 0; i < billsSize; i++) {  
 if (bills[i] == 5) {  
 five++;  
 } else if (bills[i] == 10) {  
 if (five == 0) {  
 return false;  
 }  
 five--;  
 ten++;  
 } else {  
 if (ten > 0 && five > 0) {  
 ten--;  
 five--;  
 } else if (five >= 3) {  
 five -= 3;  
 } else {  
 return false;  
 }  
 }  
 }  
 return true;  
}



第四题

int monotoneIncreasingDigits(int N) {  
 int len = 0;  
 int temp = N;  
 while (temp > 0) {  
 temp /= 10;  
 len++;  
 }  
 temp = N;  
 char str[len];  
 for (int i = len - 1; i >= 0; i--) {  
 int b=temp % 10;  
 temp/=10;  
 str[i] = 48 + b;  
 }  
 int mark = len;  
 for (int i = len - 1; i > 0; i--) {  
 if (str[i - 1] > str[i]) {  
 mark = i;  
 str[i - 1]--;  
 }  
 }  
 for (int i = mark; i < len; i++) {  
 str[i] = '9';  
 }  
 int result = 0;  
 result = atoi(str);  
 return result;  
}



第五题

int compare(const void \*a, const void \*b) {  
 int \*p1 = \*(int \*\*)a;  
 int \*p2 = \*(int \*\*)b;  
 return p1[1] - p2[1];  
}  
  
int findMinArrowShots(int\*\* points, int pointsSize) {  
 if (points == NULL || pointsSize == 0) {  
 return 0;  
 }  
 //快速排序  
 qsort(points, pointsSize, sizeof(int\*), compare);  
  
 int arrows = 1;  
 int end = points[0][1];  
  
 for (int i = 1; i < pointsSize; i++) {  
 if (points[i][0] > end) {  
 arrows++;  
 end = points[i][1];  
 }  
 }  
  
 return arrows;  
}

