ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
MÔN HỌC: HỆ THỐNG MÁY TÍNH

BÁO CÁO ĐỒ ÁN:

VIẾT CHƯƠNG TRÌNH TÍNH TỔNG VÀ KHOẢNG CÁCH CỦA 2 SỐ NGUYÊN THEO CƠ SỐ B

GIÁO VIÊN HƯỚNG DẪN:

- Thầy Thái Hùng Văn

- Thầy Đặng Trần Minh Hậu

MÃ ĐỒ ÁN: ASSIGNMENT #2.01

SINH VIÊN THỰC HIỆN:

- Trần Nguyễn Phúc Khang – 22127182

- Nguyễn Thế Trung – 22127429

1) Giới thiệu chung

Yêu cầu bài tập: Viết chương trình tính toán tổng và hiệu của

hai số nguyên X và Y ở hệ cơ số B ($\boldsymbol{X, Y \geq 0; 2 \leq B \leq 36}$) và mô tả thuật toán.

1.1) Phân tích đề bài

Dễ thấy thuật toán đơn giản nhất là chuyển về hệ cơ số 10 rồi cộng lại bằng toán tử "+" và "-" trong ngôn ngữ lập trình rồi chuyển kết quả về hệ cơ số ban đầu. Tuy nhiên đề không có chặn trên của X và Y nên X và Y có thể rất lớn khi chuyển sang hệ cơ số 10 nên nhóm đã nghĩ đến phương pháp sử dụng phương pháp xử lý bằng chuỗi (string) để thực hiện phép cộng và trừ trực tiếp trong cơ số gốc.

1.2) Hướng dẫn sử dụng chương trình

Compile file main.exe từ file file main.cpp, và thực hiện theo chỉ dẫn của chương trình để sử dụng.

2) Cài đặt thuật toán

2.1) Môi trường lập trình

• Ngôn ngữ lập trình: C++ 20.
• Compiler: g++ 13.2.0.
• IDE/Text Editor: Visual Studio Code 1.83

Các thư viện sử dụng:

• <iostream> cho phép nhập xuất cơ bản với console.
• <string> để sử dụng chuỗi để xử lí thuật toán.

2.2) Các hàm đã sử dụng

a) Hàm int charToInt():

Hàm này được sử dụng để đổi char trong số thành int. Vì xử lí trên chuỗi sẽ dễ dàng hơn xử lí trên trực tiếp số nguyên.

```
int charToInt(char c, int B){
    int res = -1;
    if(c >= '0' && c <= '9')
        return c - '0';
    else if(c >= 'A' && c <= 'Z')
        return c - 'A' + 10;
    if (res >= 0 && res < B) return res;
    throw::invalid_argument("Not an int in base B");
    return -1;
```

```
}
```

Thuật toán của hàm trên rất đơn giản được chia làm 3 trường hợp:
+ Nếu ký tự nằm trong khoảng từ ‘0’ đến ‘9’ thì kết quả sẽ bằng mã ASCII của ký tự đó trừ đi mã ASCII của ký tự ‘0’ vì mã ASCII của các ký tự từ ‘0’ đến ‘9’ liên tiếp nhau.
+ Nếu ký tự nằm trong khoảng từ ‘A’ đến ‘Z’ thì kết quả sẽ bằng mã ASCII của ký tự đó trừ đi mã ASCII của ký tự ‘A’ cộng 10 vì mã ASCII các ký tự từ ‘A’ đến ‘Z’ liên tiếp nhau và giá trị của các chữ cái bắt đầu từ 10 nên ta cộng 10 vào.
+ Tương tự với các ký tự từ ‘a’ đến ‘z’, chương trình cho phép người dùng nhập vào ký tự có thể cả in hoa lẫn in thường.
Nếu như kết quả trả về là số âm hoặc lớn hơn so với cơ số thì là người dùng đã nhập vào một ký tự không trong khoảng hợp lệ của cơ số đó, vậy ta sẽ báo lỗi và ngừng chương trình.

b) Hàm addToBaseB:

Hàm này thực hiện công việc cộng 2 số nguyên theo cơ số B.

```
string addToBaseB(string s1, string s2, int B){
    string result = "";
    int carry = 0;
    int len1 = s1.length();
    int len2 = s2.length();
    int maxLen = max(len1, len2);
    for(int i = 0; i < maxLen; i++){
        int val1 = (i < len1) ? charToInt(s1[len1 - i - 1], B) : 0;
        int val2 = (i < len2) ? charToInt(s2[len2 - i - 1], B) : 0;
        int sum = val1 + val2 + carry;
        carry = sum / B;
        sum = sum % B;
        result = char(sum + '0') + result;
    }
    if (carry > 0) {
```

```
        result = char(carry + '0') + result;
    }
    return result;
}
```

Chi tiết hàm này như sau:

1. Tạo 1 biến result để lưu kết quả và biến carry để lưu số nhớ.

2. Cho vòng lặp chạy từ đầu tới cuối chuỗi số.
3. Sau đó chúng em sẽ lấy kí tự cuối ra (tức hàng đơn vị) của mỗi số nguyên để cộng lại với nhau. Nếu cộng lại mà lớn hơn cơ số B, thì sẽ lấy kết quả vừa cộng được chia dư cho B và lưu vào 1 biến nhớ mà chúng em đã gọi là biến carry. Biến carry này sẽ được cộng tiếp theo vào lần cộng của chữ số kế tiếp.

c) Hàm subtractToBaseB:

Hàm này thực hiện công việc trừ 2 số nguyên theo cơ số B

```
string subtractToBaseB(string s1, string s2, int B){
    string result = "";
    if(!isGreater(s1, s2)) return "-" + subtractToBaseB(s2, s1, B);
    int carry = 0;
    int len1 = s1.length();
    int len2 = s2.length();
    int maxLen = max(len1, len2);
    int borrow = 0;
    for (int i = 0; i < maxLen; i++) {
        int value1 = (i < len1) ? charToInt(s1[len1 - i - 1], B) : 0;
        int value2 = (i < len2) ? charToInt(s2[len2 - i - 1], B) : 0;
        value1 -= borrow;

        if (value1 < value2) {
            value1 += B;
            borrow = 1;
```

```
        } else {
            borrow = 0;
        }
        int diff = value1 - value2;
        char ch = char(diff + '0');
        result = ch + result;
    }
    while (result.length() > 1 && result[0] == '0') {
        result = result.substr(1);
    }
    return result;
}
```

Chi tiết hàm này như sau:
1.Bởi vì thuật toán này chỉ áp dụng được khi số bị trừ lớn hơn số trừ nên nếu số bị trừ bé hơn số trừ, ta sẽ gọi đệ quy đảo vị trí của số trừ và số bị trừ để được kết quả (nạp chồng toán tử “>” để so sánh sẽ được nói cụ thể ở dưới) rồi thêm dấu ‘-’ vào để kết quả xuất ra có dấu ‘-’ phía trước biểu diễn giá trị âm.

2. Tương tự như phép tính tổng, ta tạo 1 biến result để lưu kết quả trả về và biến carry để lưu lại số nhớ do thực hiện phép trừ có nhớ, biến này khỏi tạo với giá trị 0.
3. Lặp từ phần tử đầu tiên đến phần tử cuối cùng của số dài hơn và thực hiện phép trừ (do chúng ta lưu theo chữ số hàng đơn vị nằm ở ngoài cùng bên trái thay vì viết chữ số hàng đơn vị ở ngoài cùng bên phải như khi nhập vào hay viết trên giấy):
• Chữ số thứ i ở biến kết quả sẽ bằng chữ số thứ i của số bị trừ trừ đi số nhớ và trừ đi chữ số thứ i của số trừ (hoặc không trừ nếu i lơn hơn chiều dài của số trừ).
• Nếu kết quả của phép tính trên âm thì ta phải mượn 1 từ vị trí kế tiếp nên chữ số thứ i của kết quả được cộng thêm giá trị của cơ số và để biến nhớ thành 1, nếu không mượn thì biến nhớ là 0.
• Do kết quả có thể ngắn hơn số bị trừ nên thực hiện xoá các số 0 dư thừa trong kết quả.

d) Hàm int isGreater():

Hàm này để so sánh coi số nào lớn hơn, bởi vì có thể số X sẽ lớn hơn số Y. Chúng em dùng hàm này để so sánh để gọi đệ quy đảo ngược để ta luôn có số lớn trừ số bé.

```
bool isGreater(const std::string& s1, const std::string& s2) {
    int len1 = s1.length();
    int len2 = s2.length();
    if (len1 < len2) {
        return false;
    } else if (len1 > len2) {
        return true;
    }
    for (int i = 0; i < len1; ++i) {
        if (s1[i] < s2[i]) {
            return false;
        } else if (s1[i] > s2[i]) {
            return true;
        }
    }
    return false;
}
```

Cụ thể thuật toán như sau:
• Nếu độ dài 2 số khác nhau thì số nào dài hơn sẽ lớn hơn.
• Nếu độ dài 2 số bằng nhau thì xét từng chữ số, bắt đầu từ chữ số cách xa hàng đơn vị nhất cho đến chữ số ở hàng đơn vị. Với mỗi chứ số thứ i, nếu hai chữ số thứ i của 2 số khác nhau thì số nào lớn hơn sẽ lớn hơn, nếu bằng nhau thì tiếp tục xét chứ số kế tiếp.
• Nếu chạy đến hết hàng đơn vị mà vẫn chưa trả về kết quả thì 2 số bằng nhau nên trả về false.

e) Hàm main:

Xử lí tất cả các công việc còn lại, từ cho người dùng nhập vào 2 số nguyên cho đến việc xét tính hợp lệ của cơ số B.

```
int main(){
    int B;
    string X, Y;

    cout << "Enter base (2-36): ";
    cin >> B;
    cin.ignore();
    if (B < 2 || B > 36) {
        cout << "Invalid base. Please enter a valid base between 2 and 
36." << endl;
        return 1;
    }
    cout << "Enter first number X in base " << B << " (X >= 0): ";
    cin >> X;

    cout << "Enter second number Y in base " << B << " (Y >= 0): ";
    cin >> Y;

    string sum = addToBaseB(X, Y, B);
    string difference = subtractToBaseB(X, Y, B);
    cout << "Sum: " << sum << endl;
    cout << "Difference: " << difference << endl;
    return 0;
}
```