

**DANH SÁCH THÀNH VIÊN:**

1612018 TRẦN QUỐC ANH  
1612036 PHẠM PHÚ QUỐC BẢO  
1612041 PHAN HẢI BÌNH  
1612082 NGUYỄN HỒNG ĐĂNG  
1612175 LÂM TỀ HÀO

**Môn học:** KIẾN TRÚC MÁY TÍNH & HỢP NGỮ  
**Giảng viên:** LÊ VIẾT LONG

Đại học Quốc gia TP.HCM – ĐH Khoa học Tự nhiên

BÁO CÁO ĐỒ ÁN 1

BIỂU DIỄN VÀ TÍNH TOÁN SỐ HỌC TRÊN MÁY TÍNH

khoa cntt – ĐH KHTN – ktmt & HN

March 28, 2018

**MỤC LỤC**

[1. Phân công công việc & mức độ hoàn thành: 2](#_Toc510039462)

[Bảng phân công: 2](#_Toc510039463)

[Mức độ hoàn thành: 2](#_Toc510039464)

[2. Phạm vi biểu diễn của dữ liệu: 3](#_Toc510039465)

[3. nội dung chương trình: 3](#_Toc510039466)

[1. Cấu trúc bigint: 3](#_Toc510039467)

[2. Hàm main: 3](#_Toc510039468)

[3. Hàm nhập, xuất: 3](#_Toc510039469)

[4. Các toán tử +, -, \*, /: 3](#_Toc510039470)

[5. Các toán tử AND, OR, XOR, NOT: 4](#_Toc510039471)

[6. CÁC TOÁN TỬ DỊCH >>, <<: 4](#_Toc510039472)

[7. Các PHÉP đổi hệ: 4](#_Toc510039473)

[8. giao diện: 5](#_Toc510039474)

[4. Testcase và giao diện trên cmd: 8](#_Toc510039475)

[5. các nguồn tham khảo: 10](#_Toc510039476)

## Phân công công việc & mức độ hoàn thành:

### Bảng phân công:

| Thành viên | Công việc |
| --- | --- |
| 1612018 - TRẦN QUỐC ANH | * Thiết kế hàm main. * Thiết kế hàm nhập, xuất dữ liệu. * Thiết kế kiểu dữ liệu. |
| 1612036 - PHẠM PHÚ QUỐC BẢO | * Thiết kế giao diện (WinForm). * Kiểm lỗi chương trình. |
| 1612041 - PHAN HẢI BÌNH | * Các hàm tính toán +, -, \*, /. |
| 1612082 - NGUYỄN HỒNG ĐĂNG | * Các hàm dịch bit >>, <<. * Các hàm AND, OR, XOR, NOT |
| 1612175 - LÂM TỀ HÀO | * Các hàm chuyển đổi kiểu dữ liệu. * Các hàm chuyển đổi hệ số. |

### Mức độ hoàn thành:

|  |  |
| --- | --- |
| Nội dung | Mức độ hoành thành |
| Hàm nhập: | 100 % |
| Hàm xuất: | 100 % |
| Hàm chuyển hệ 10 sang 2: | 100 % |
| Hàm chuyển hệ 2 sang 10: | 100 % |
| Hàm chuyển hệ 2 sang 16: | 100 % |
| Hàm chuyển hệ 10 sang 16: | 100 % |
| Các toán tử +, -, \*, /: | 100 % |
| Các toán tử AND, OR, XOR, NOT: | 100 % |
| Toán tử dịch trái, phải <<, >>: | 100 % |

*Có giao diện chương trình, thực hiện bằng WinForm.*

* **Mức độ hoàn thành tổng thể:** 100%

## Phạm vi biểu diễn của dữ liệu:

**Giá trị nhỏ nhất:** *(hệ 10)*

**Giá trị lớn nhất:** *(hệ 10)*

*Đối với hệ 2: 128 bit.*

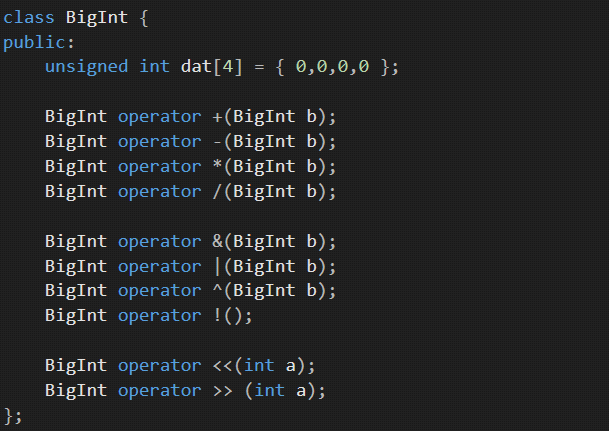
*Đối với hệ 16: 32 kí tự.*

## nội dung chương trình:

### Cấu trúc bigint:

Dùng để lưu trữ 1 số nguyên lớn (16 byte): dữ liệu sẽ được lưu trữ trong 1 mảng gồm 4 phần tử mang kiểu dữ liệu *unsigned int*, mỗi phần tử là 4 byte.

Dùng class tạo đối tượng kiểu dữ liệu *BigInt*:



### Hàm main:

Có sử dụng tham số dòng lệnh (cmd) với cách thức chạy chương trình: *<.exe> <.txt> <.txt>* với file .exe là tên chương trình, 2 file .txt lần lượt là file input và output.

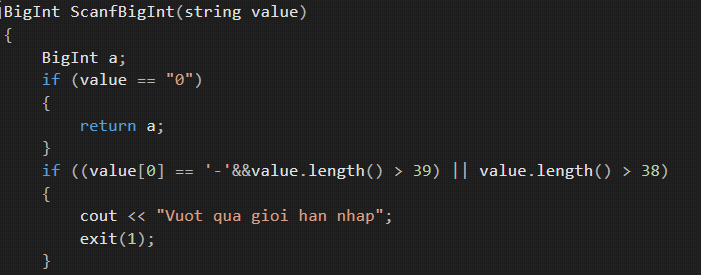
Chương trình sẽ đọc file .txt đầu vào và xử lí bài toán, hàm có chức năng kiểm tra đề bài, nếu file đầu vào sai thì sẽ báo lỗi vào file đầu ra trên dòng tương ứng với yêu cầu và tiếp tục xét dòng kế tiếp của file đầu vào. Nếu đề bài đúng cú pháp thì sẽ phân tích đề bài (theo từng dòng của file) và xử lí theo yêu cầu của mỗi dòng file, sau đó xuất kết quả ra file đầu ra theo kết quả tương ứng với từng dòng của file đầu vào.

### Hàm nhập, xuất:

* Số nhập vào sẽ là ở dạng chuỗi:

Chuỗi nhập đó sẽ được xử lí và quy về chuỗi nhị phân 128 bit (do struct BigInt là 1 số nguyên lớn 16 byte), sau đó dãy bit đó sẽ được chia làm 4 chuỗi 32 bit và lần lượt từng chuỗi 32 bit này sẽ được đổi sang hệ 10 và lưu vào struct BigInt. Với mỗi chuỗi 32 bit tương ứng với mỗi số *int* trong mảng dữ liệu của cấu trúc.

**Bước 1**: kiểm tra số nhập vào có phải giá trị 0 hoặc nằm ngoài giới hạn



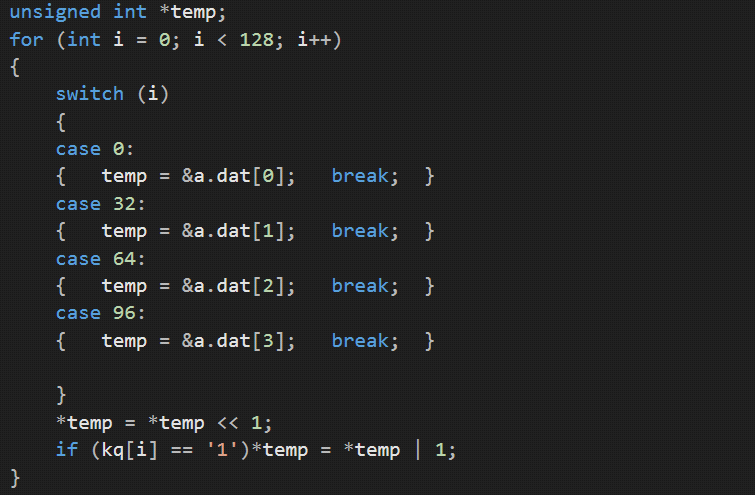
**Bước 2**: Đổi chuỗi nhập vào thành mảng nhị phân char[] có 128 phần tử gồm kí tự 0 hay 1. Nếu là sô âm thì sẽ sử dụng dạng bù 2.



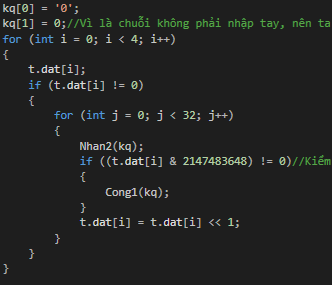
Thông qua hàm Chia2 thao tác đối với số nguyên lớn ở dạng chuỗi( ví dụ chuỗi nhập vào là "123", sau khi gọi hàm Chia2 thì kết quả thu được là số dư "1" trong phép chia chuỗi "123" cho 2 và chuỗi lúc này được gán giá trị là thương "61") , kết quả của hàm Binary(value, kq) sẽ trả về chuỗi nhị phân dựa số dư là 1 hay 0 ở mỗi lần chia 2 của số nguyên lớn nhập vào ban đầu. Quá trình sẽ kéo dài cho đến khi nào chuỗi số nguyên lớn có giá trị là "0". Thuật toán lấy ý tưởng từ việc tìm ra dãy nhị phân cho một số nguyên từ thực tế, áp dụng vào lập trình đối với số nguyên lớn thông qua việc xử lý chuỗi.

Trong trường hợp số nguyên lớn nhập vào là số âm, thì ta sẽ dùng phương thức số bù 2. Nghĩa là từ chuỗi kết quả thu được sau quá trình Chia2 thành công, ta đổi chuỗi bit đó sang dạng bù 2.

**Bước 3**: Xác định vị trí phần tử trong BigInt và gán bit trong chuỗi value vào:



* Xuất giá trị của *BigInt* ra:



Xây dựng hàm giải quyết số âm sẽ làm chức năng chuyển bit của số âm dạng bù 2 thành số không dấu, áp dụng cùng phương pháp như đã dùng trong hàm Binary của quá trình nhập BigInt.

Để in ra được một số nguyên lớn như vậy, ta không thể in trực tiếp biến đó được, mà phải thông qua việc in một chuỗi. Tương tự như việc đọc dữ liệu vào, ta sẽ sử dụng hàm Nhan2 để xử lý số nguyên lớn với dạng chuỗi. Duyệt lần lượt từng biến dat trong mảng 4 phần tử của BigInt, ở mỗi lần duyệt, ta kiểm tra vị trí bit 1 trong tất cả 32 bit của một biến dat. Mỗi lần gặp vị trí bit 1 thì ta cộng thêm 1 vào chuỗi số nguyên lớn, với mỗi lần duyệt từng vị trí như vậy, ta dùng hàm Nhan2 để nhân 2 giá trị của chuỗi số nguyên lớn lên.

Khi đã duyệt xong hết và xử lý với dữ liệu số âm thì kết quả thu được là số BigInt cần tìm.

### Các toán tử +, -, \*, /:

**Sơ lược:**

* Thực hiện các phép + - \* / ở dạng chuỗi thập phân. Đầu tiên đổi các số từ BigInt sang chuỗi thập phân, xử lý và đổi lại BigInt.
* Tất cả phần xử lý đều dựa vào nguyên lý + - \* / cơ bản hồi Tiểu học.

**Phần xử lý:**

* + - Toán tử +, -:

Xét các số hạng, nếu chỉ có 1 số âm thì phép toán là phép trừ, nếu 2 số âm thì sẽ là phép cộng nhưng có biến đánh dấu để kết quả trả về mang giá trị âm.

Tiến hành thêm các số 0 vào đầu chuỗi số ngắn hơn sao cho độ dài 2 chuỗi bằng nhau.

Tiến thực hiện phép tính theo từng cặp kí tự từ phải sang trái đến khi kết thúc. Nếu kết quả của cặp kí tự lớn hơn 9 (đối với phép cộng) hoặc nhỏ hơn 0 (đối với phép trừ) thì lưu biến nhớ là 1 và tiến hành trừ 10 kết quả đó (đối với phép cộng) hoặc cộng 10 (đối với phép trừ), biến nhớ sẽ được xử lý vào kết quả của cặp kí tự xử lý sau.

Kết quả là số âm thì sẽ có thêm dấu “-“ ở vị trí đầu chuỗi số.

* Toán tử \*:

Thực hiện xét dấu, nếu chỉ có 1 số âm thì kết quả trả về sẽ âm (dùng 1 biến để đánh dấu), còn lại sẽ là dương.

Thực hiện phép nhân tuần tự số này với các kí tự của số kia (dùng một mảng chuỗi động để lưu). Nếu được nhân với kí tự phải cùng thì không cần phải thêm 0 ở cuối, còn lại thì sẽ thêm 0 ở cuối nếu cứ qua trái 1 kí tự.

Thực hiện cộng tất cả số ở mảng chuỗi ở bước 2 (dùng phép cộng đã được viết trước ở trên) sẽ ra được kết quả tạm thời.

Thực hiện thêm dấu “-“ ở vị trí đầu chuỗi số dựa vào biến đánh dấu ban đầu để có được kết quả cuối cùng.

* Toán tử /:

Thực hiện xét dấu, nếu chỉ có 1 số âm thì kết quả trả về sẽ âm (dùng 1 biến để đánh dấu), còn lại sẽ là dương.

Xét nếu số bị chia nhỏ hơn số chia thì kết quả sẽ là 0.

Xét nếu số chia là 0 thì kết quả trả về là dấu “-“ để đánh dấu.

Hạ dần dần các chữ số của số bị chia cho đến khi lớn hơn số chia (gắn số đó vào biến a).

Thực hiện phép chia số a với số bị chia. Thực hiện bằng cách trừ số a với số chia cho đến khi số a nhỏ hơn số chia thì dừng lại. Số lần thực hiện phép trừ chính là kết quả của phép chia đó. Số a còn lại chính là số dư.

Tiếp tục hạ 1 chữ số ở số bị chia xuống và thực hiện tiếp phép chia xuống cuối số a. Nếu số a nhỏ hơn số chia thì kết quả của phép chia đó là 0 và số a cũng chính là số dư. Ngược lại, chúng ta thực hiện chia như bước 5.

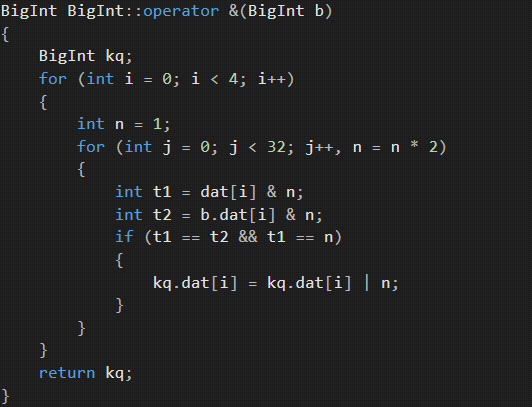
Lặp lại bước sau cho tới khi số bị chia không còn để hạ. Khi đó chuỗi kết quả của phép chia lần lượt chính là kết quả tạm thời.

Thực hiện thêm dấu “-“ ở vị trí đầu chuỗi số dựa vào biến đánh dấu ban đầu để có được kết quả cuối cùng.

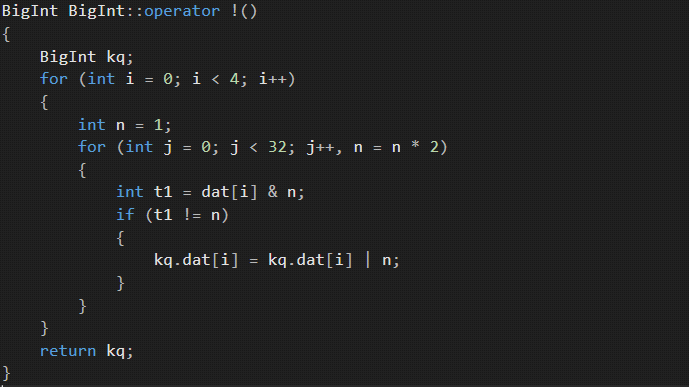
### Các toán tử AND, OR, XOR, NOT:

Với AND, OR, XOR: lần lượt xét cặp bit cùng vị trí với nhau từ 2 số, xét từ bit cuối lên bit đầu, sau đó trả kết quả về tùy theo mỗi phép toán AND, OR, XOR. Tiếp tục thực hiện đến bit đầu tiên, kết quả của cặp bit sau sẽ nằm ở phía bên trái kết quả cặp bit đầu.

Cụ thể: ta có 2 biến tạm (t1 và t2) để lưu trữ từng cặp bit tương ứng của 2 số (t1 và t2 sẽ được lấy bằng cách & với 2n). Tùy từng phép toán áp dụng vào t1 và t2 sẽ có kết quả bit tương ứng. Sau đây là ví dụ cho phép toán AND. (Tương tự với các phép toán còn lại)



Với NOT: đổi các bit 0 thành 1 và ngược lại.



### CÁC TOÁN TỬ DỊCH >>, <<:

Với <<: xét giá trị đầu của phần tử cuối cùng trong mảng dữ liệu của cấu trúc. Nếu giá trị đó là 1 thì tạo biến tạm có giá trị là 1 (và ngược lại) và tiến hành dịch bit. Sau đó xét tiếp phần tử tiếp theo và +1 vào nếu biến tạm khi nãy có giá trị 1. Tiếp tục quá trình cho đến hết 4 phần tử dat.

Với >>: xét giá trị cuối cùng của phần tử đầu trong mảng dữ liệu của cấu trúc. Nếu giá trị đó là 1 thì tạo biến tạm có giá trị 1 (và ngược lại) và tiến hành dịch bit. Phần tử tiếp theo sẽ cộng thêm cho 231 nếu biến tạm khi nãy có giá trị 1. Tiếp tục quá trình cho đến hết 4 phần tử dat.

### Các PHÉP đổi hệ:

* 10 và 2 *(dec & bin)*:

Hệ 10 -> 2: lấy từng giá trị của các biến *unsigned int* trong *BigInt*, sau đó đổi về hệ 2 bằng cách thức chia dư thông thường rồi ghép 4 chuỗi (tương đương 4 giá trị lại với nhau) và trả về 1 chuỗi 128 bit. (4 chuỗi bit trên được khởi tạo 32 bit 0)

Hệ 2 -> 10: chuỗi bit tham số sẽ có định dạng là 128 bit, chuỗi này sẽ được chia làm 4 chuỗi 32 bit, mỗi chuỗi 32 bit sẽ quy đổi về hệ 10 và lần lượt đưa vào vị trí của mảng dữ liệu trong cấu trúc.

* 16 và 10 *(hex & dec)*:

Hệ 16 -> 10: tham số là 1 chuỗi kí tự, bắt đầu xử lí từ kí tự phải cùng của chuỗi, mỗi kí tự sẽ quy ra giá trị nhất định rồi dùng công thức chuyển hệ để tính toán và đưa vào phần tử cuối cùng trong mảng dữ liệu, nếu kết quả này vượt quá số 268435456 thì kết quả tiếp theo sẽ được lưu sang phần tử kế của mảng dữ liệu. Cứ làm vậy cho đến khi kết thúc chuỗi (đến kí tự đầu của chuỗi).

Hệ 10 -> 16: làm tương tự như đổi từ hệ 10 -> hệ 2, kết quả trả về sẽ là chuỗi kí tự. Định dạng gốc sẽ có 32 kí tự, với 32 kí tự được khởi tạo toàn “0”.

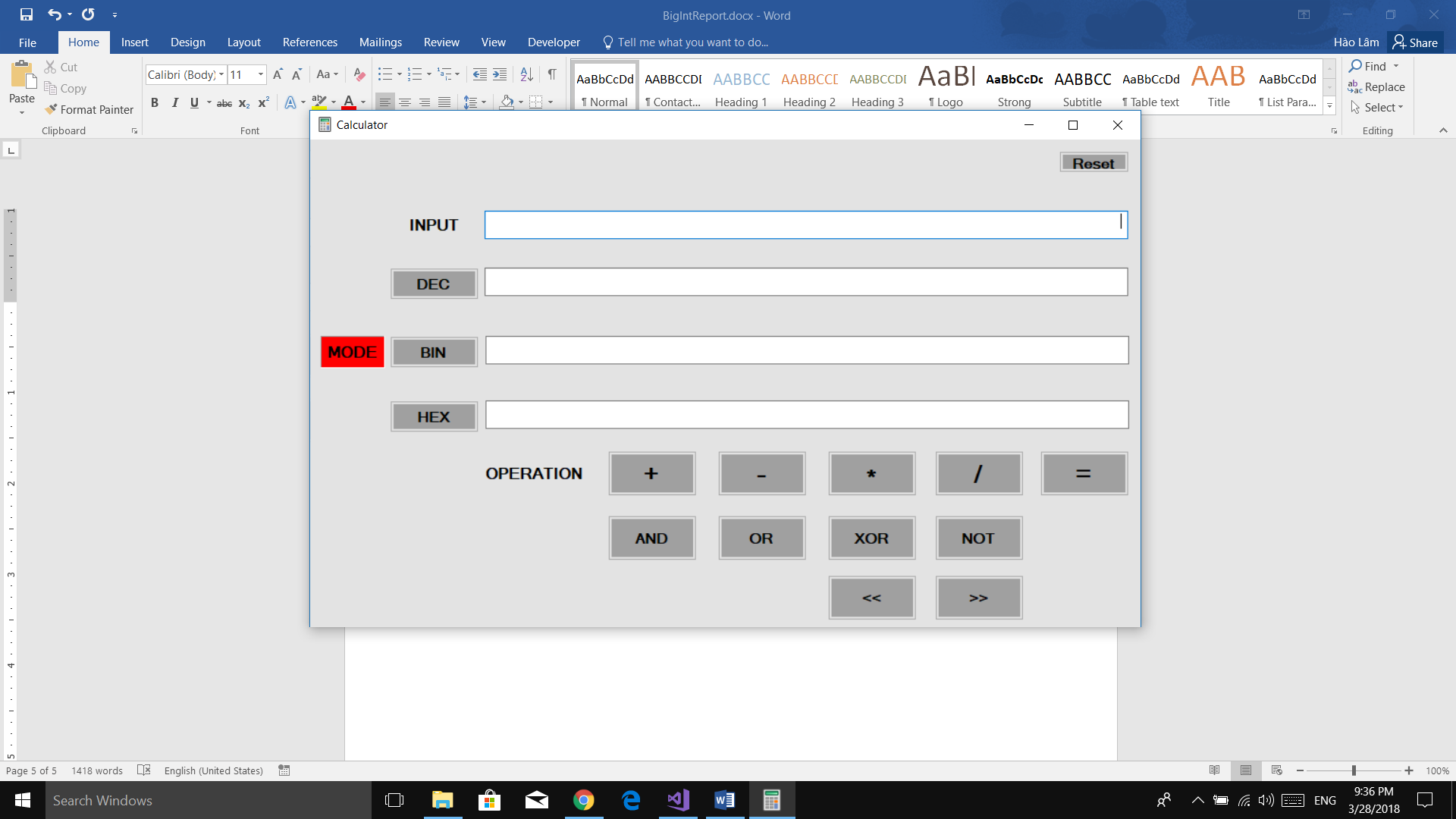
* 16 và 2 *(hex & bin)*:

Hệ 16 -> 2: thực hiện đổi hệ gián tiếp: từ hệ 16 -> 10, sau đó từ hệ 10 -> 2, trả về chuỗi bit.

Hệ 2 -> 16: thực hiện tương tự đổi từ hệ 2 -> 10.

### giao diện:

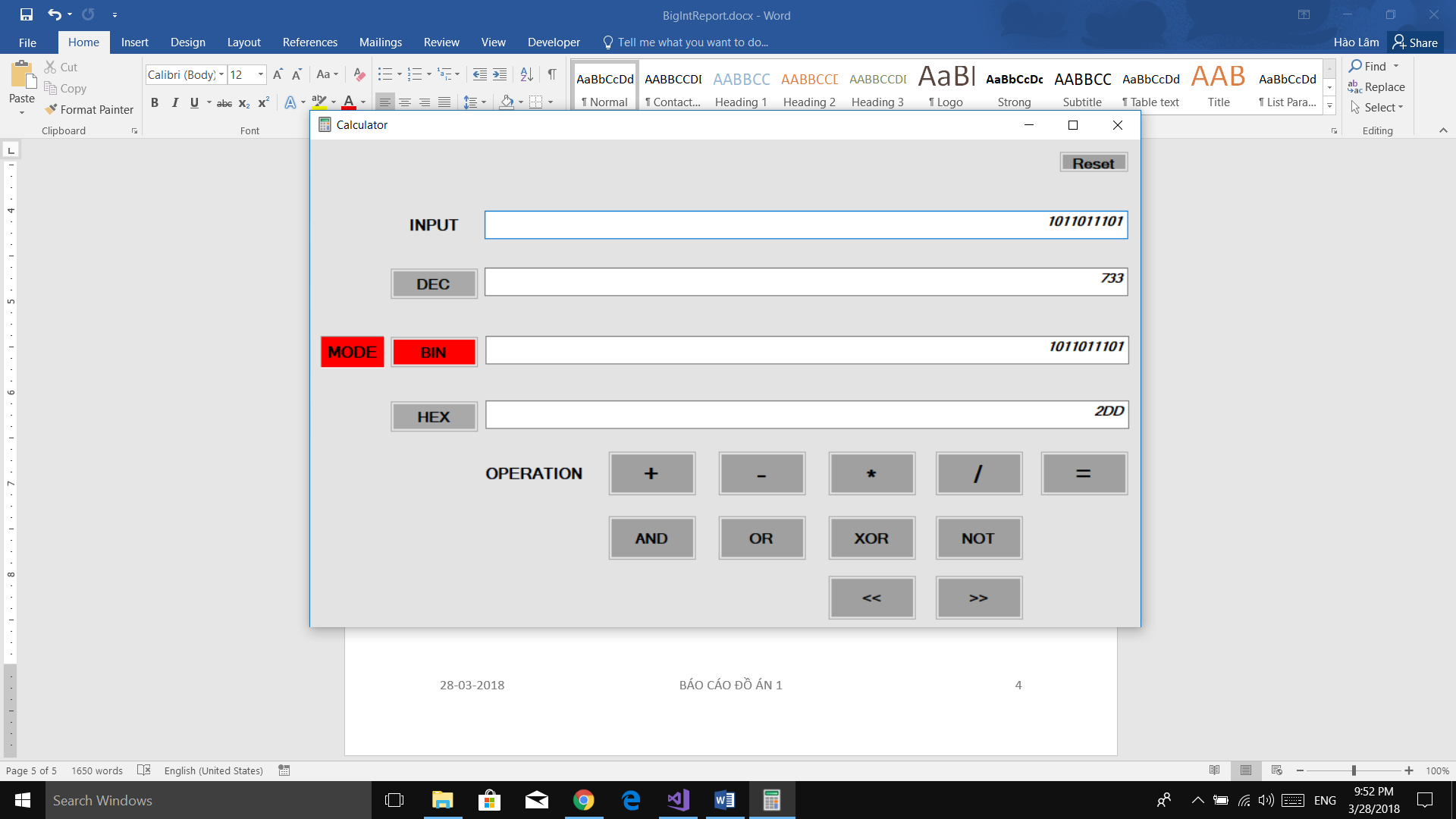
Được lập trình bằng WinForm. Giao diện tương tự như máy tính thông thường.

Người dùng sẽ nhập số đầu vào cùng với hệ số chọn trước và yêu cầu thực hiện phép tính, kết quả trả về sẽ được hiển thị trên cả 3 hệ *bin*, *dec*, *hex*.

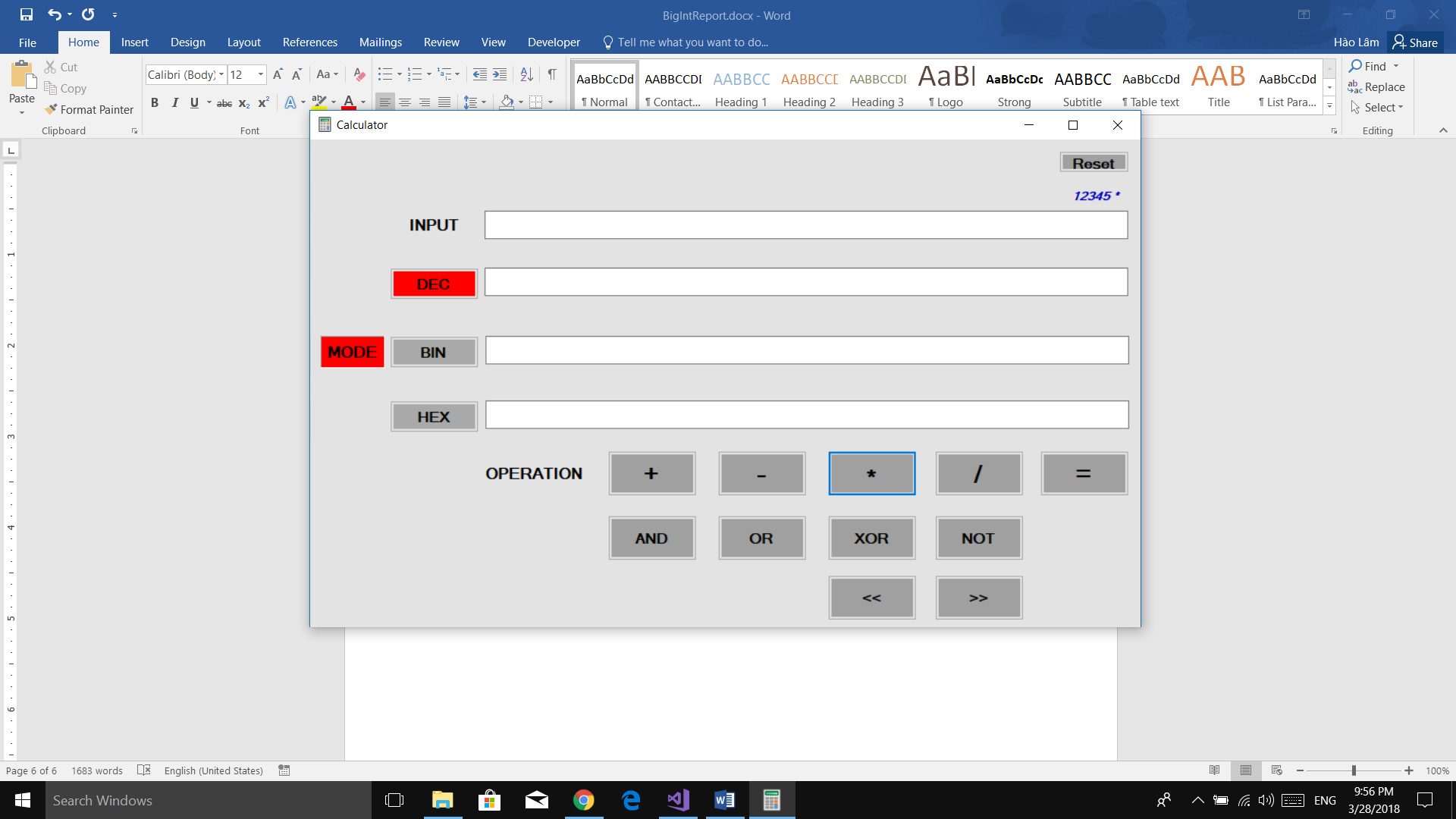
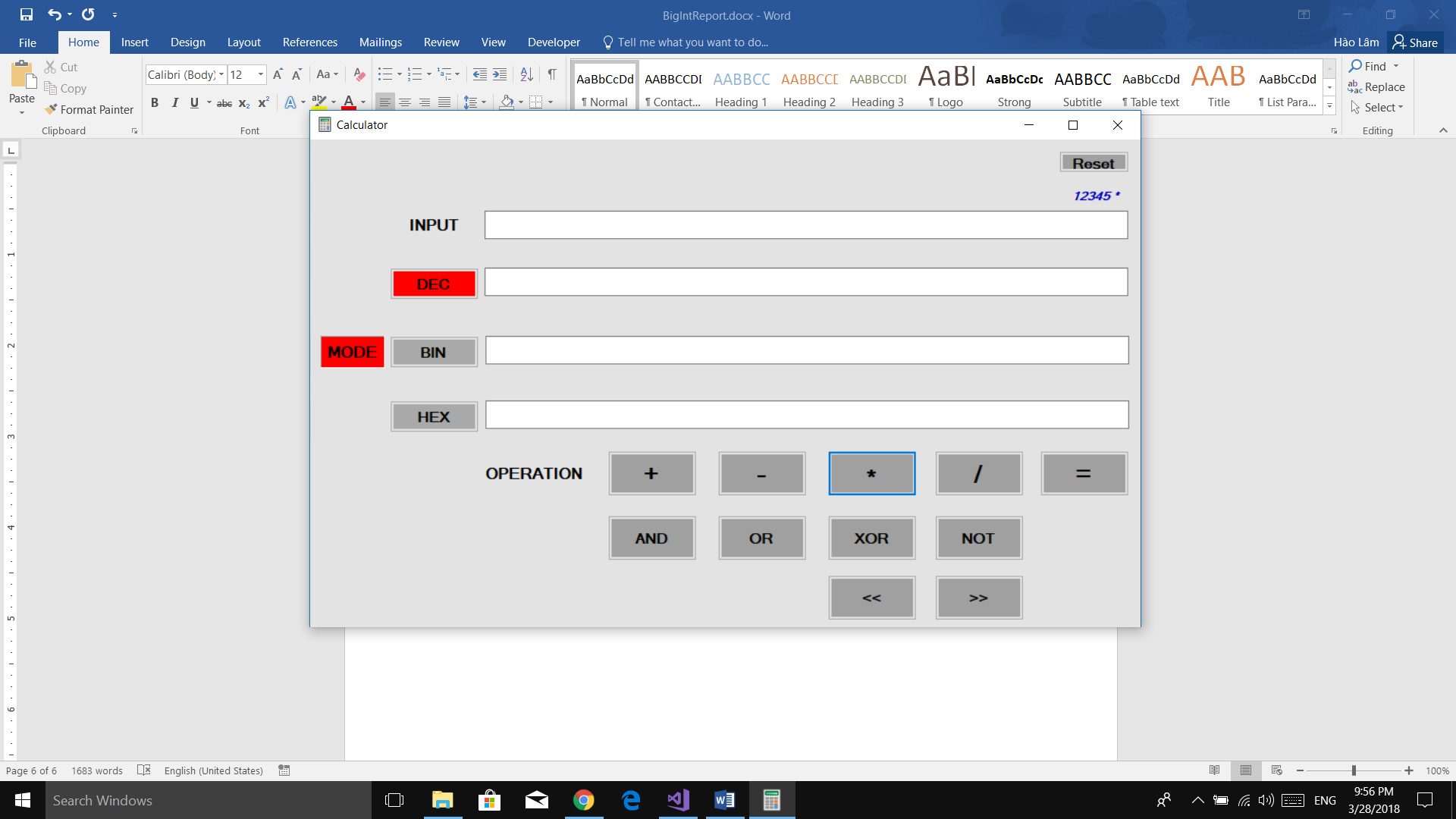
Hình 1: chọn 1 trong 3 chế độ nhập để nhập liệu

Cú pháp nhập:

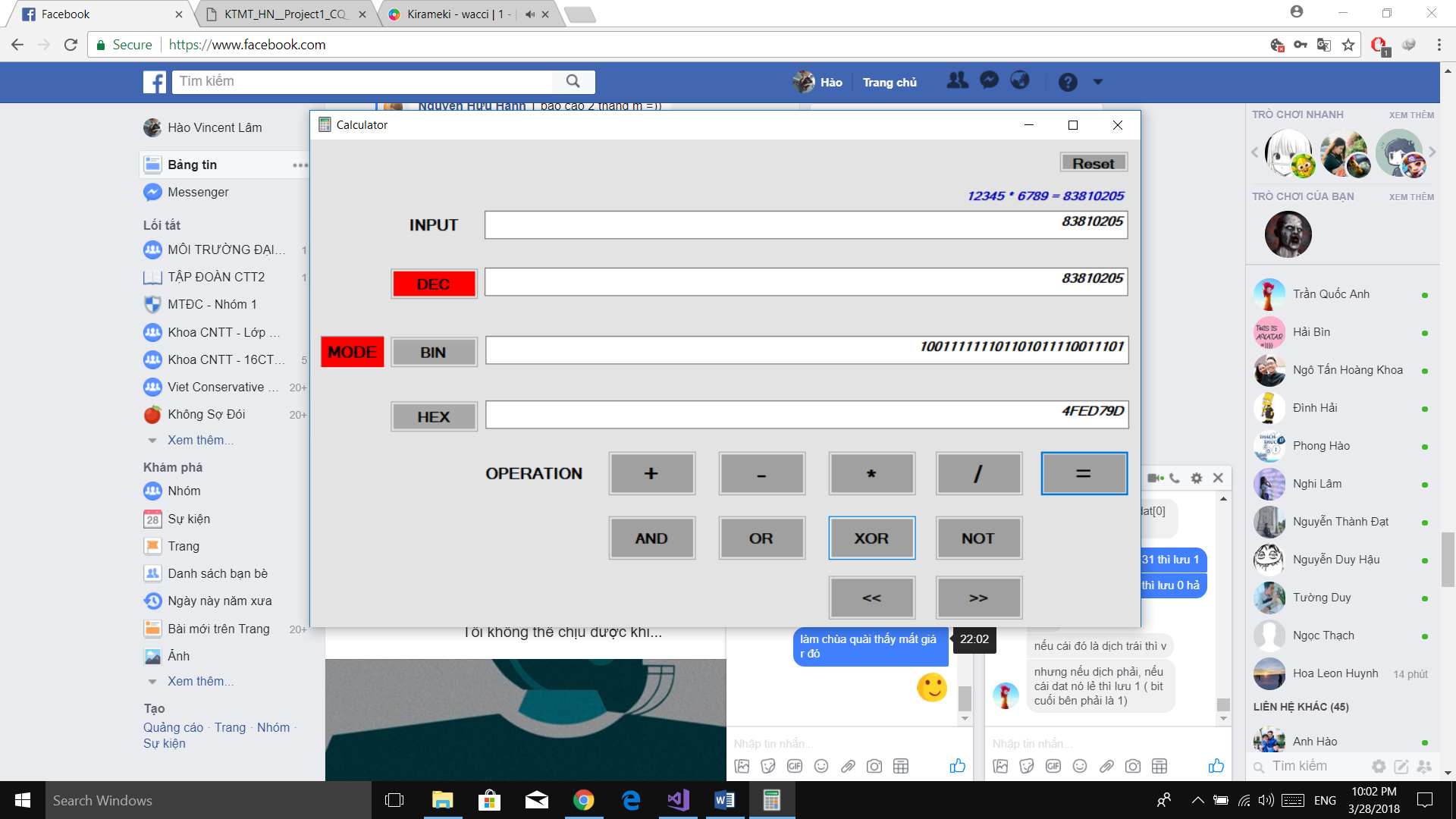
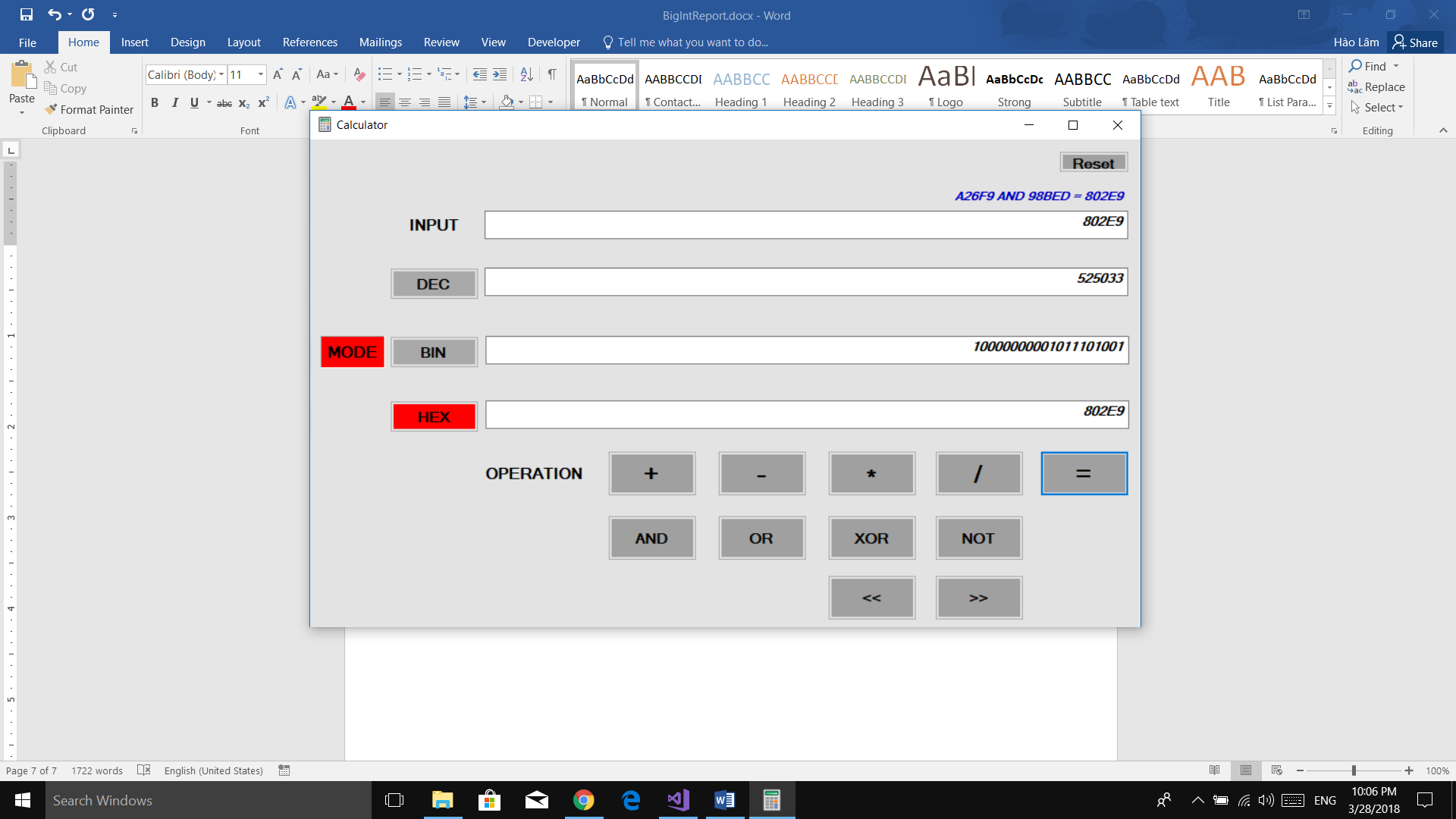
* Với hệ số 10: nếu nhập kí tự không phải là số thì sẽ thoát chương trình.
* Với hệ số 2: nếu nhập khác 0 hoặc 1 thì thoát chương trình.
* Với hệ số 16: với kí tự chữ cái phải nhập in hoa và giới hạn kí tự từ A -> F, nếu nhập chữ cái thường hoặc ngoài miền giá trị thì sẽ thoát chương trình.

Số đã nhập vào sẽ được biểu hiện dưới các hệ số khác theo hàng tương ứng.

Hình 2: Kết quả hiện thị khi nhập số hệ 2

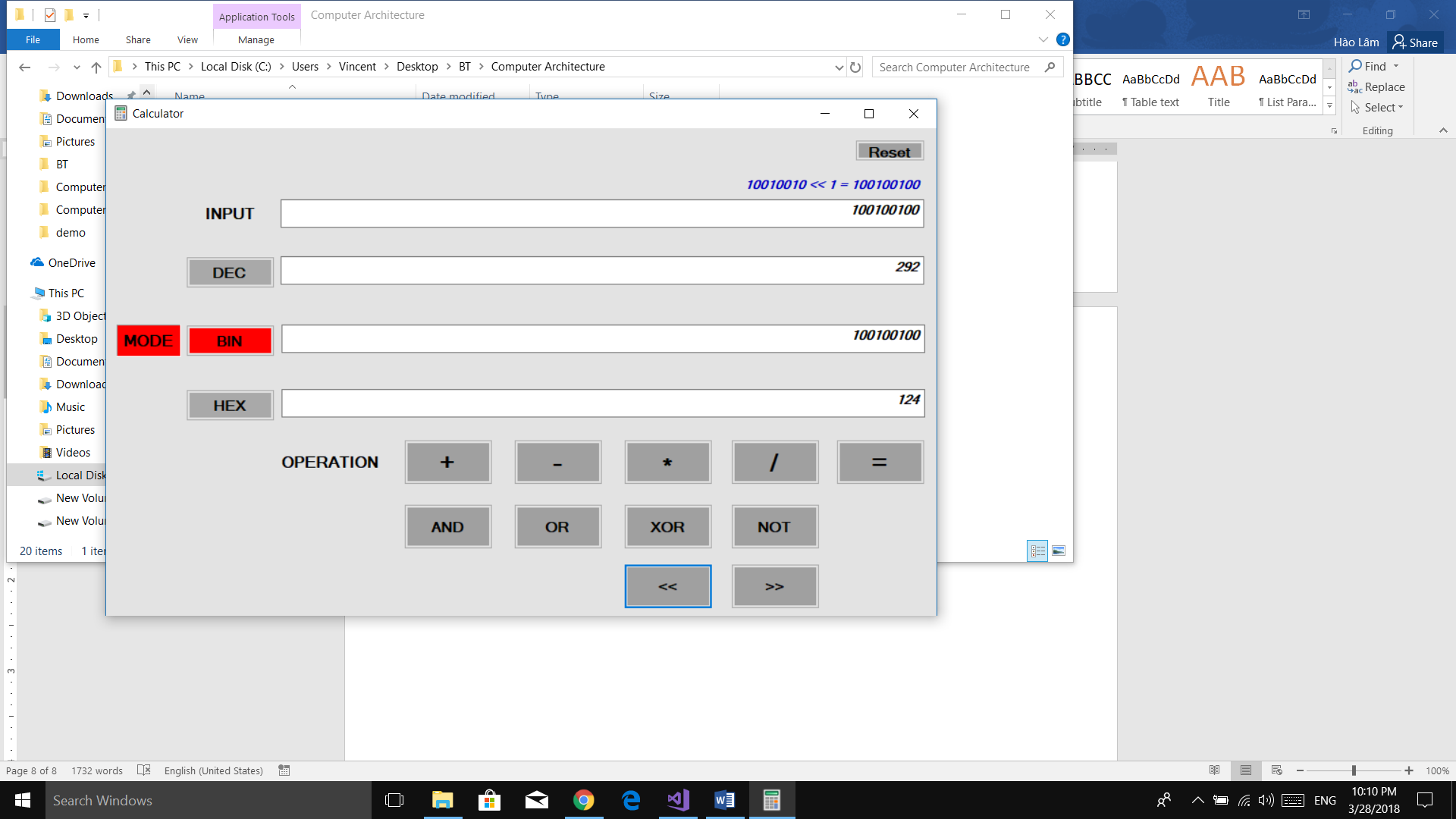
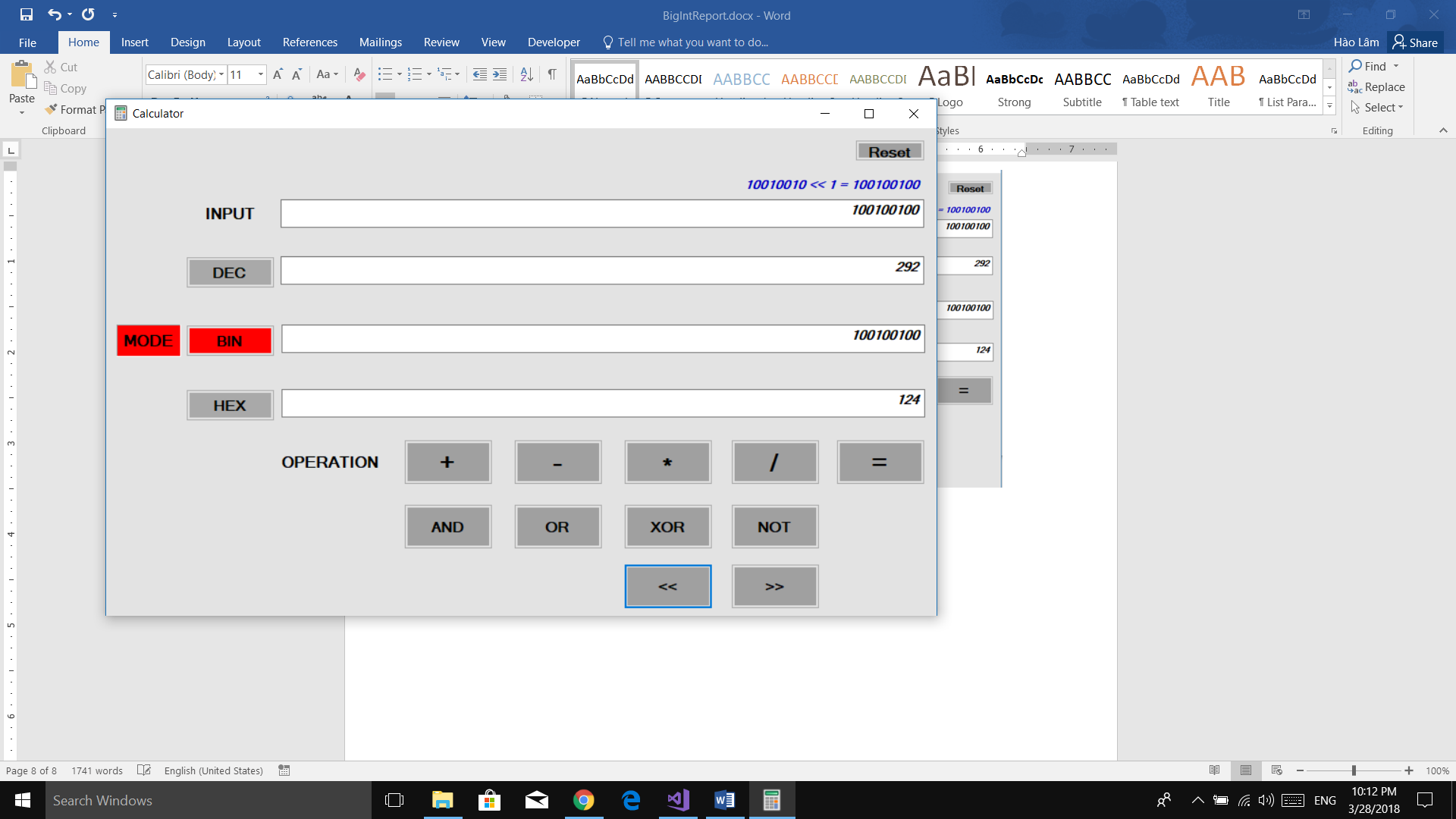
Nếu muốn thực hiện phép tính, sau khi nhập xong số đầu, chỉ cần chọn phép tính và nhập tiếp số sau rồi chọn dấu để trả về kết quả.

Hình 3: Nhập vào số hệ 10 12345 sau đó chọn phép \*



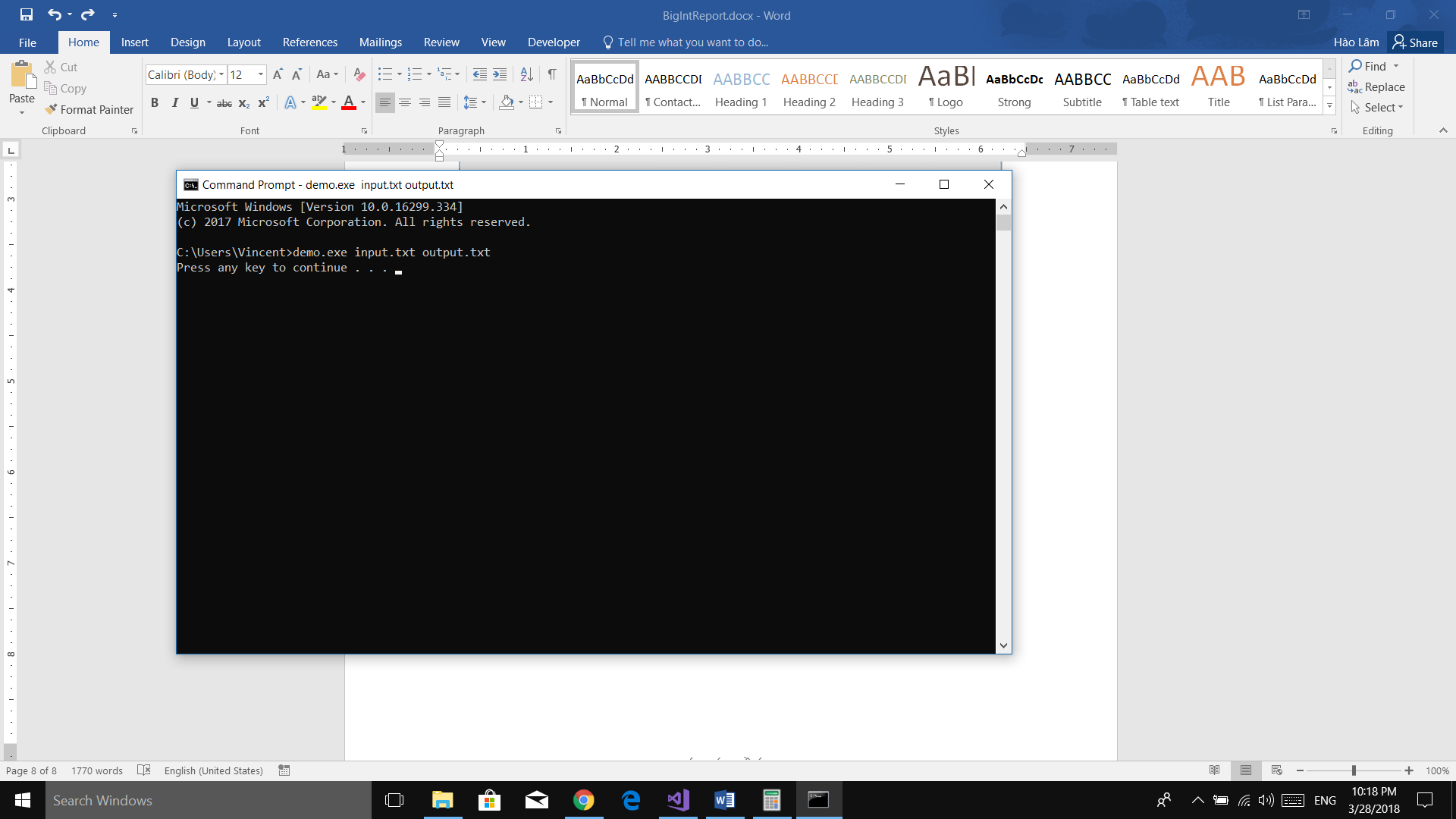
Hình 5: Kết quả phép tính A26F9 AND 988BED (16)

Hình 4: Nhập tiếp số 6789 sau đó chọn dấu "=" sẽ trả về kết quả phép tính

Nút  dùng để xóa hết dữ liệu đã nhập.

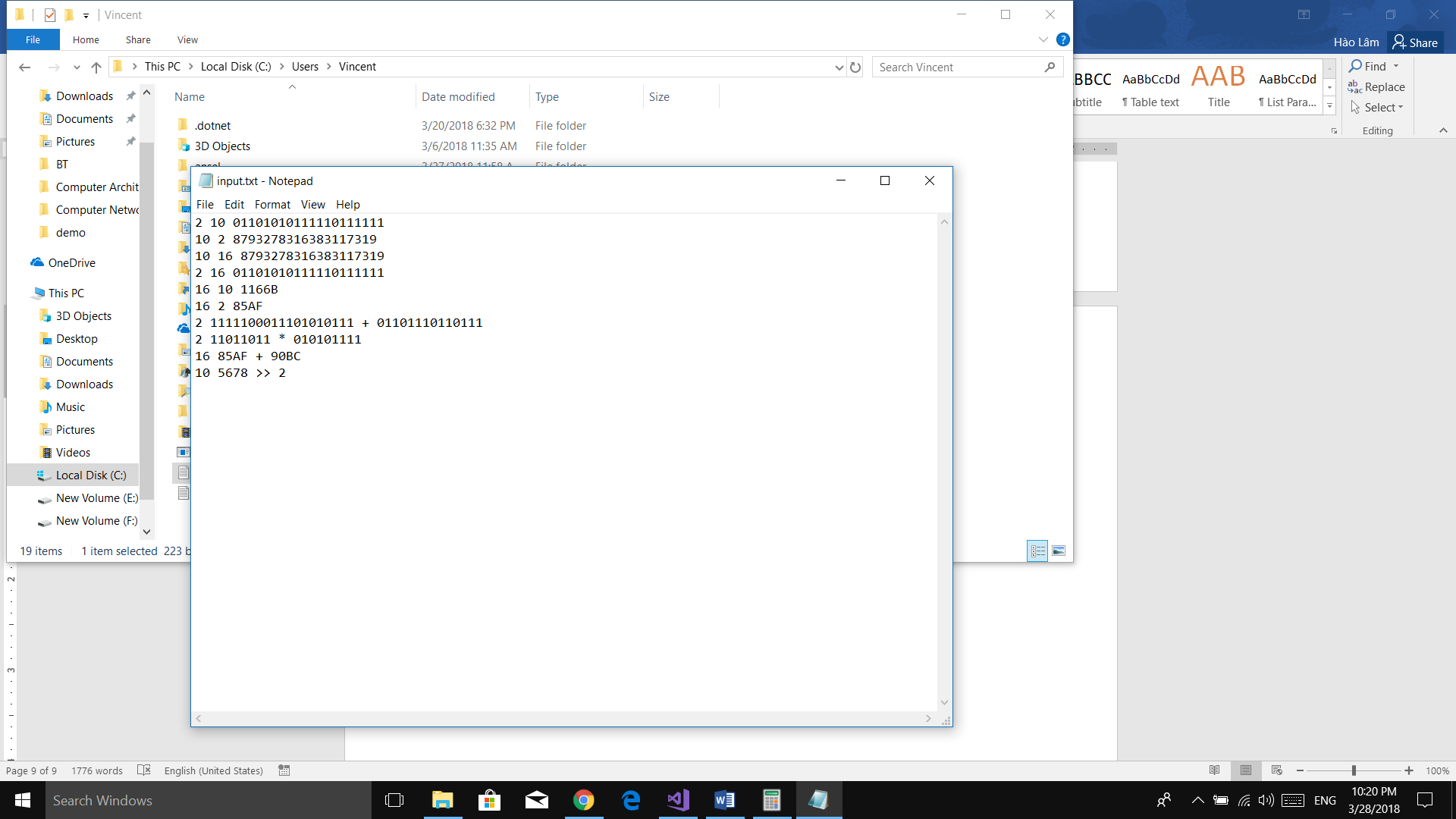
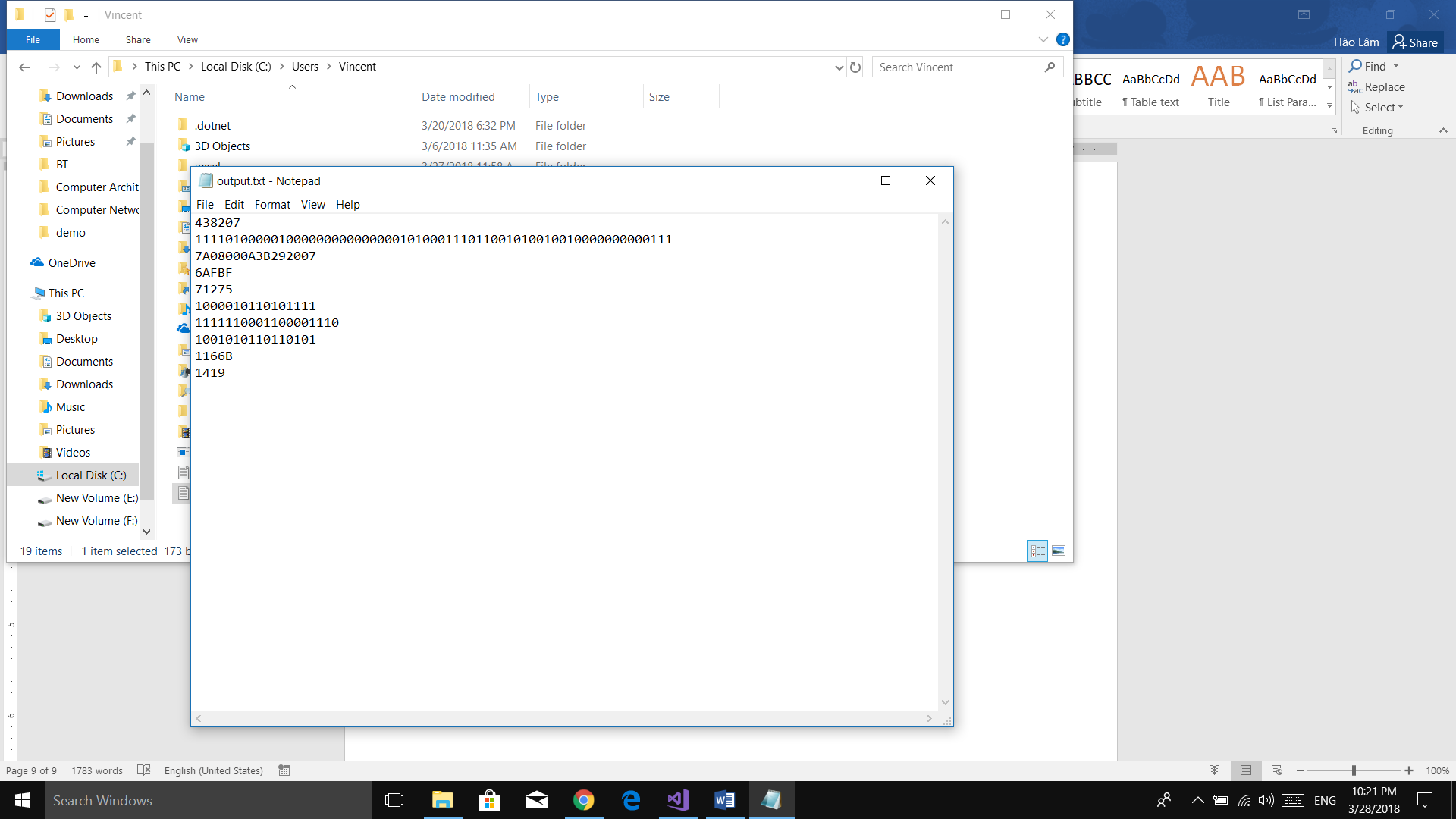
Hình 6: dịch trái số nhị phân 10010010

## Testcase và giao diện trên cmd:

Gọi thực hiện chương trình với file đầu vào *input.txt* và đầu ra *output.txt*:

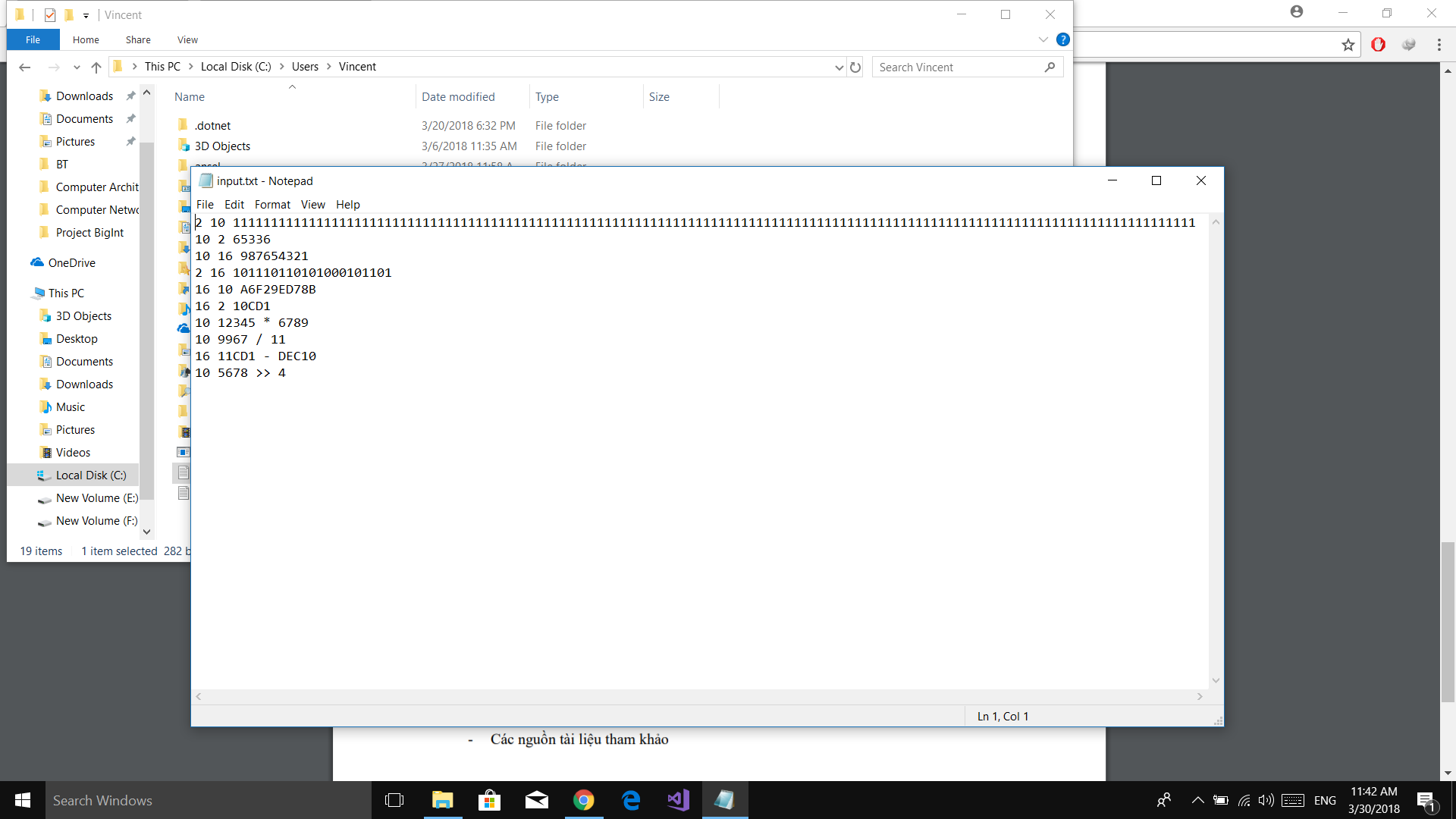
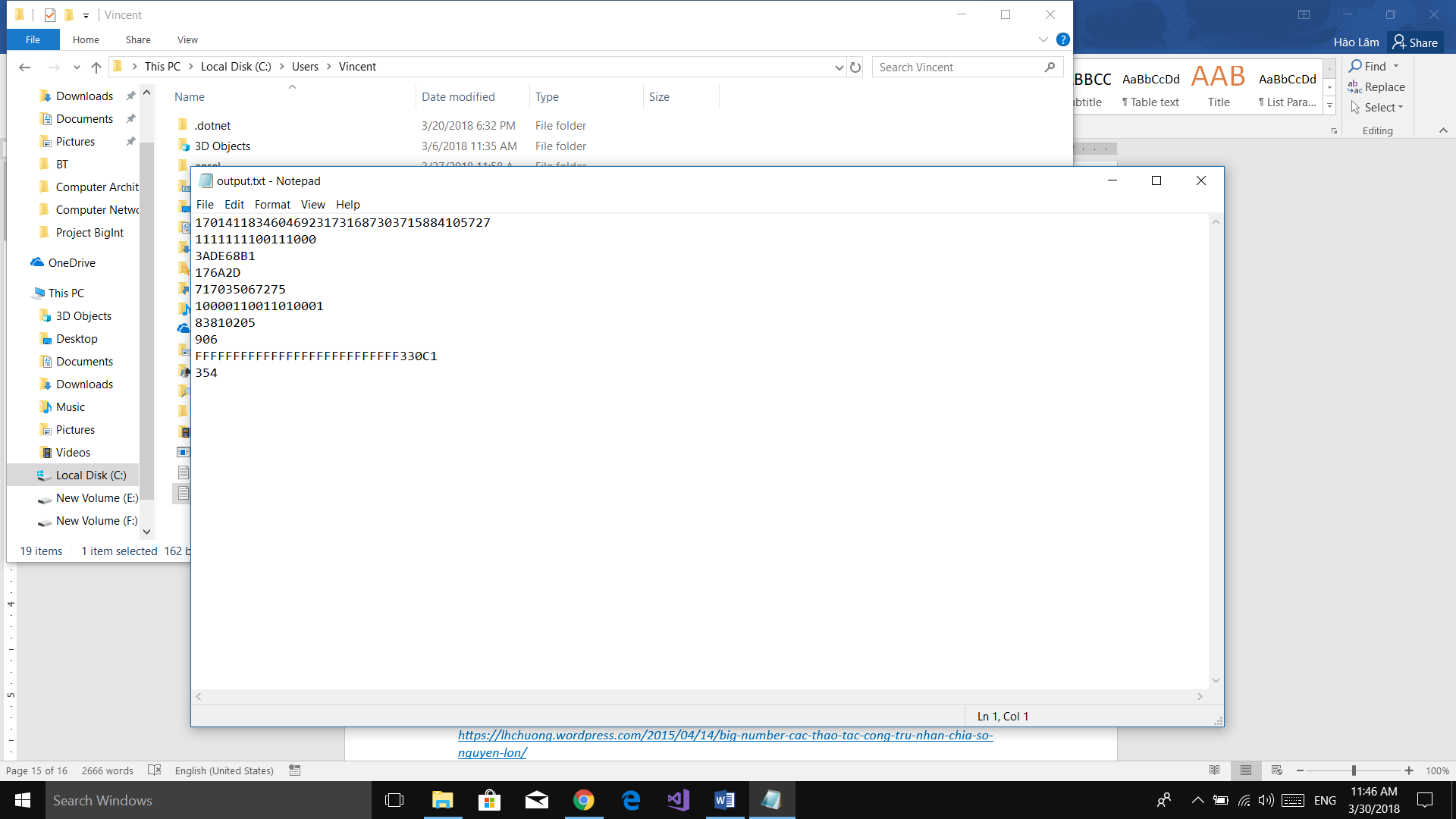
Hình 7:Chương trình chạy thành công

* **Testcase 1:**



Hình 9: Kết quả tập tin đầu ra *output.txt*

Hình 8: Tập tin đầu vào *input.txt*

* **Testcase 2:**

Hình 11: Tập tin đầu ra output.txt

Hình 10: TẬP TIN ĐẦU VÀO INPUT.TXT

## các nguồn tham khảo:

[*https://stackoverflow.com/questions/946813/c-cli-converting-from-systemstring-to-stdstring*](https://stackoverflow.com/questions/946813/c-cli-converting-from-systemstring-to-stdstring)

[*https://lhchuong.wordpress.com/2015/04/14/big-number-cac-thao-tac-cong-tru-nhan-chia-so-nguyen-lon/*](https://lhchuong.wordpress.com/2015/04/14/big-number-cac-thao-tac-cong-tru-nhan-chia-so-nguyen-lon/)

[*http://voer.edu.vn/m/bu-2-he-nhi-phan/2a7ddbde*](http://voer.edu.vn/m/bu-2-he-nhi-phan/2a7ddbde)

HẾT