**Họ và Tên: Trần Thị Ngọc Diệp**

**MSSV: 1827005**

**Lớp: B2HK182 – Cấu Trúc Dữ Liệu và Giải Thuật**

**(TUT 4 & LAB 4)**

------------------------

**TUT 4**

**PART 1: STACK**

**Question 1:**

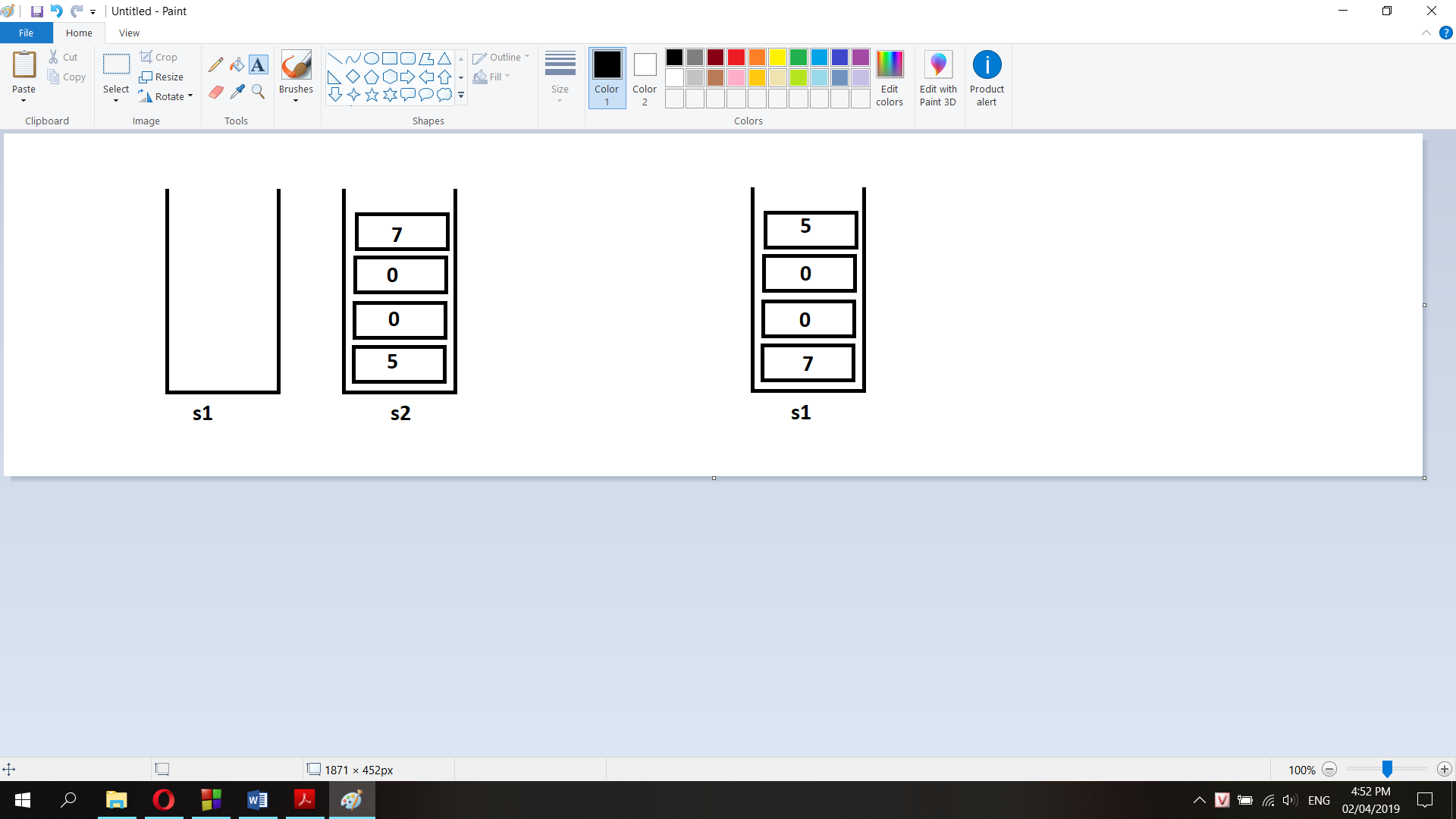
a1 = 7

a2 = 0

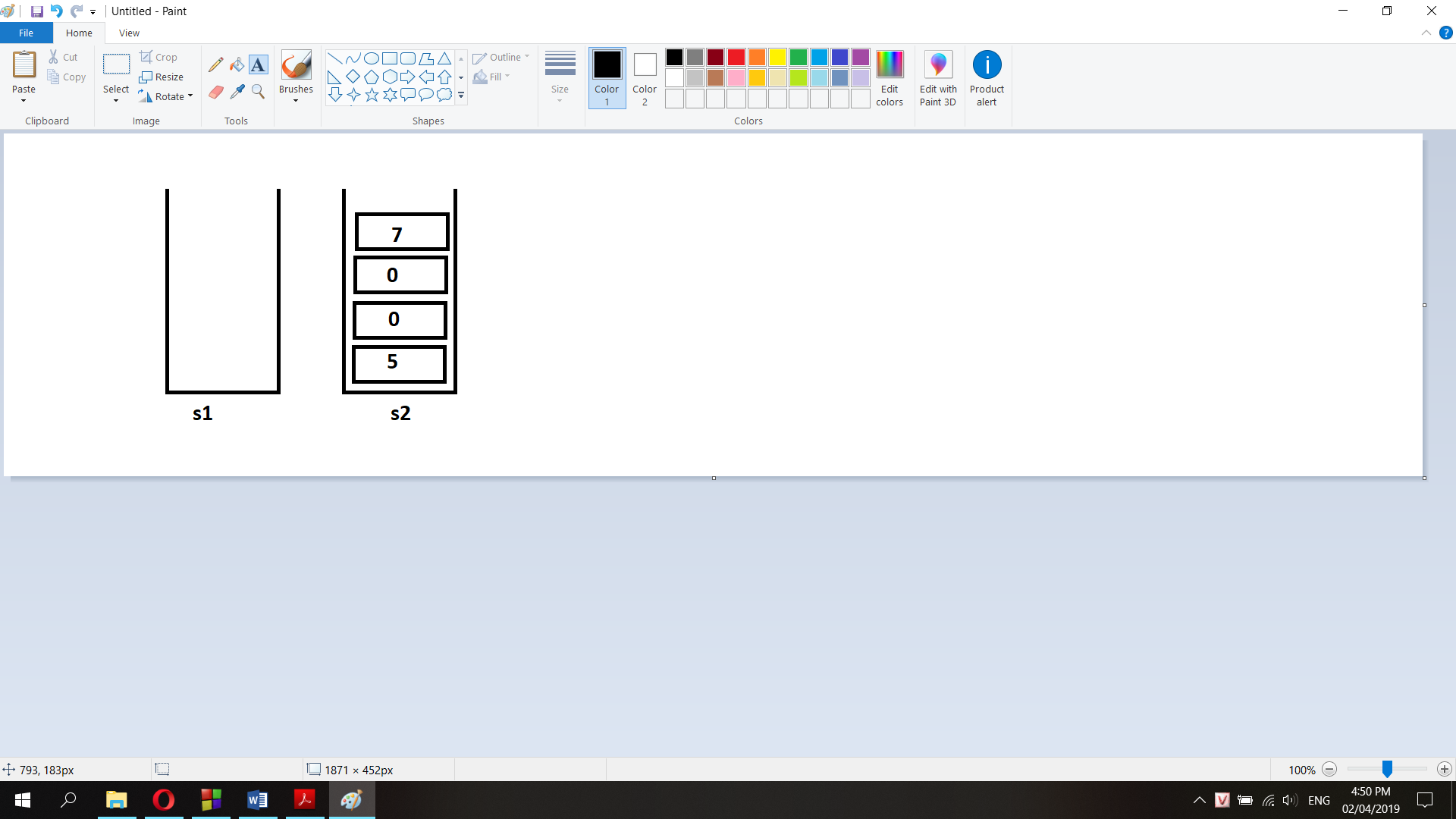
a3 = 0

a4 = 5

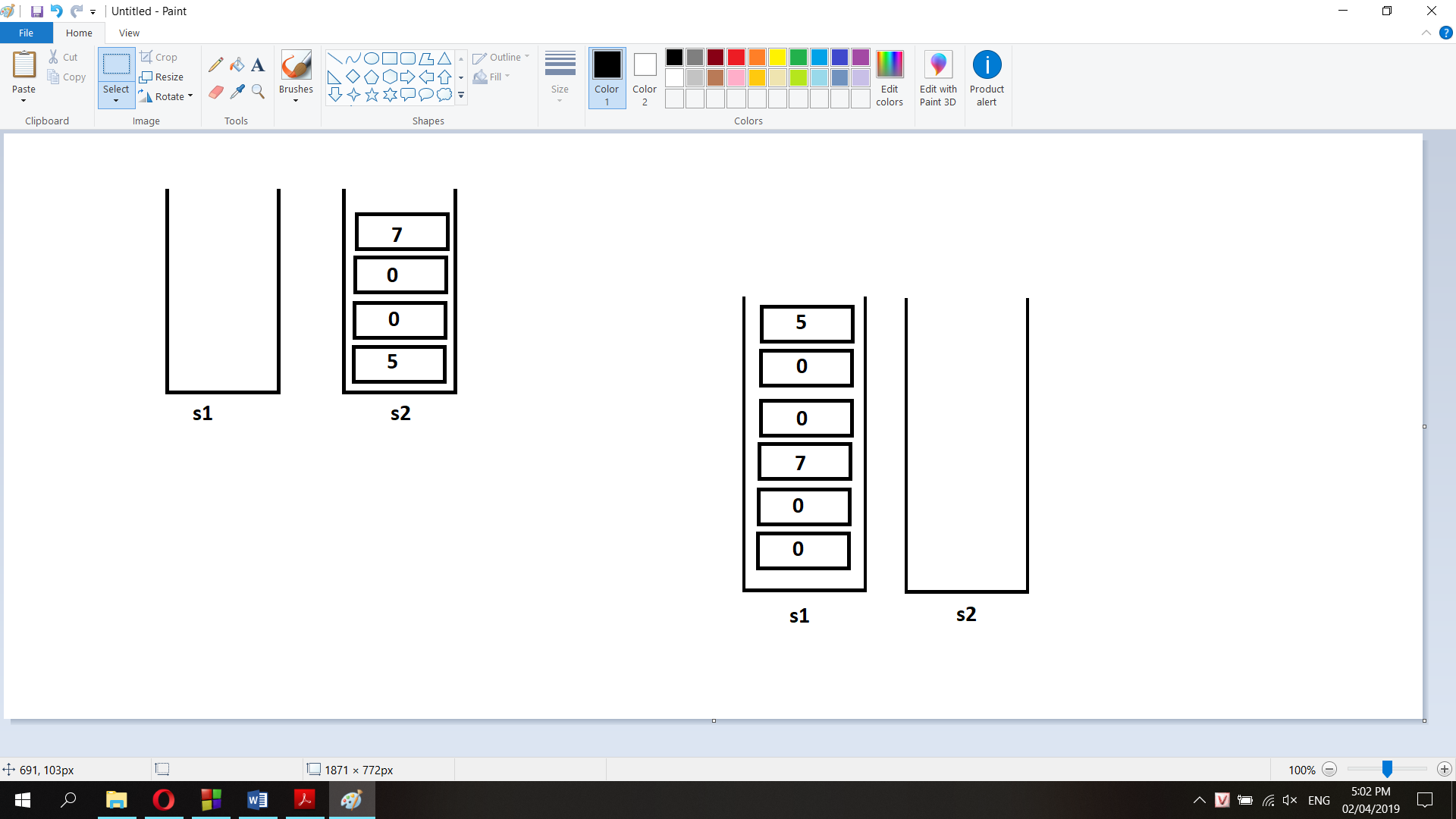
* Stack s1 sau khi push các phần tử vào (stack s2 vẫn rỗng):



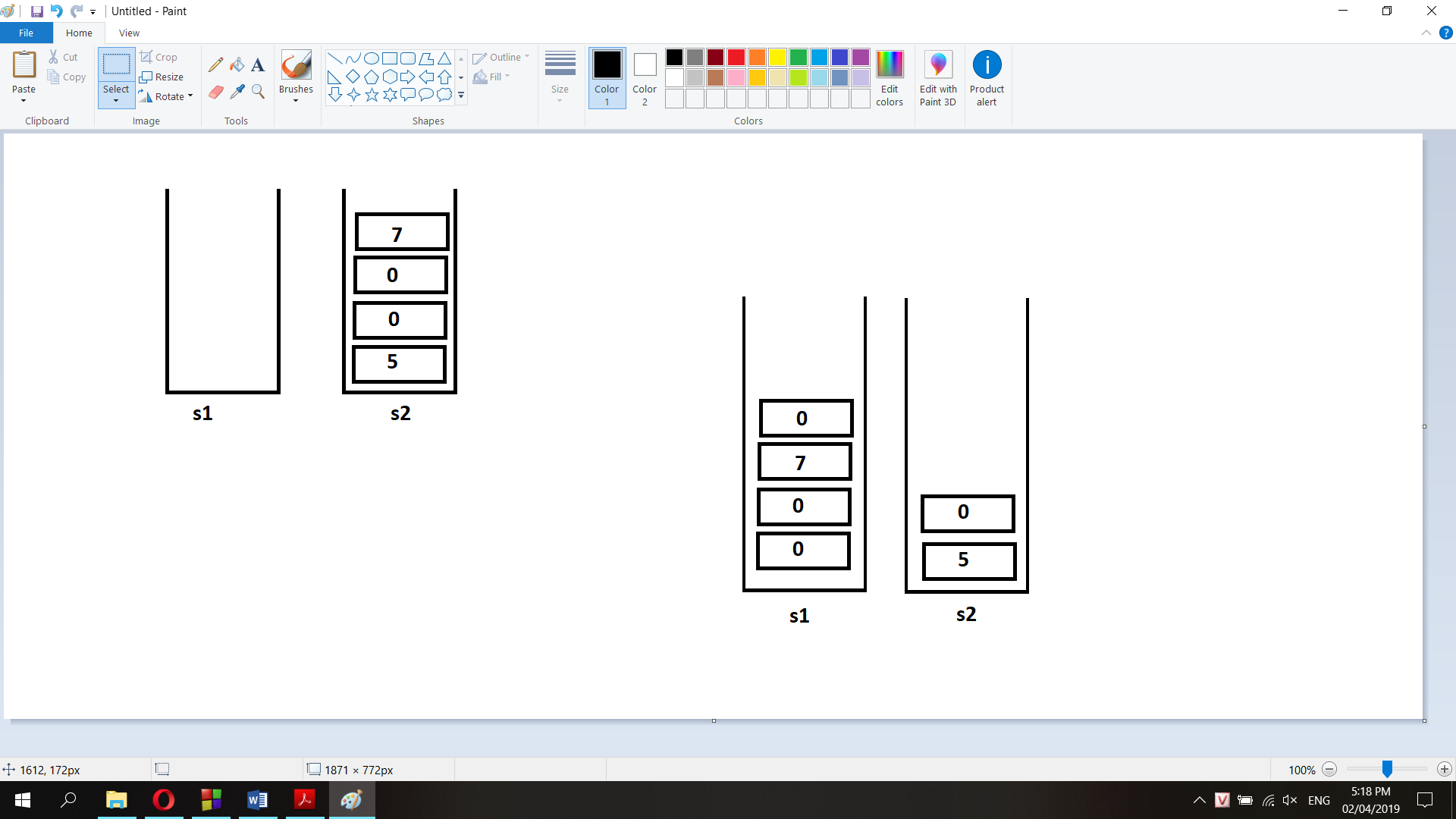
* **Line 8**: Vòng lặp while chuyển lần lượt các phần tử của stack s1 sang stack s2 cho đến khi stack s1 empty.



* **Line 14**: Sau khi push 2 giá trị 0 và 0 vào s1, vòng lặp while tiếp tục push lần lượt các giá trị của stack s2 vào stack s1 cho đến khi stack s2 empty.



* **Line 18:** 4 câu lệnh cuối lần lượt lấy 2 phần tử trên cùng của stack s1 và push vào s2:



**Question 2:**

**a. Thay đổi dòng 6 và 7 để in ra 4 số cuối của MSSV:**

while (!EmptyStack (s1)){

PopStack (s1, x);

PushStack(s3,x);

}

while (!Empty Stack(s3){

PopStack (s3,x);

cout<<x;

PushStack(s1,x);

}

while (!Empty Stack(s1){

PopStack(s1,x);

PushStack(s2,x);

}

**b.** Đoạn code trên với mục đích là giữ nguyên giá trị 2 stack sau câu lệnh 6 và 7 như đoạn code nguyên gốc.

**Question 3:** This algorithm removes the N-th element in the sourceStack. The order of the

remaing elements must be preserved after the removal.

RemoveN (ref sourceStack<Stack>, val N <data>)

Stack Q;

Int i = 0;

while (i <count – N) {

PopStack (sourceStack, x);

PushStack (Q, x);

i++;

}

PopStack (sourceStack, x);

while (!EmptyStack(Q)) {

PopStack (Q, x);

PushStack(SourceStack, x);

}

end RemoveSecond

**PART 2: Queue**

**Question 4: Student ID: 1827005**

**Loop 2**: push tất cả các giá trị của MSSV vào queue Q1, front trỏ vào số đầu tiên (1).

**Loop 4**: push lần lượt các giá trị trong Q1 sang stack S (1->8->2->7), cho đến khi gặp số 0 thì chạy loop 4.2.2, loop này tính tổng giá trị các phần tử có trong stack S, sau đấy push giá trị này vào Q2. Loop 4 tiếp tục chạy cho đến khi Q1 rỗng. Cuối cùng trong stack S có 1 phần tử giá trị 5, Q1 rỗng và Q2 có 1 phần tử có giá trị 18 (7+2+8+1).

**Question 5:**

**a1 = 2**

**a2 = 7**

**a3 = 0**

**a4 = 0**

**a5 = 5**

**Bảng giá trị input:**

|  |  |  |
| --- | --- | --- |
| Line | File | Value |
| 1 | **a1 + a2** | **9** |
| 2 | **a2 + a3** | **7** |
| 3 | **a3 + a4** | **0** |
| 4 | **a3 – a4** | **0** |
| 5 | **a4 + a5** | **5** |
| 6 | **a5 + a1** | **7** |
| 7 | **a2 – a3** | **7** |
| 8 | **a1 + a2 – a3** | **9** |
| 9 | **a2 – a3 + a4** | **7** |
| 10 | **a3 + a4 + a5** | **5** |
| 11 | **a1 – a2** | **-5** |
| 12 | **a1 \* a2** | **14** |
| 13 | **a2 \* a3** | **0** |
| 14 | **a3 \* a4** | **0** |
| 15 | **a1 + a2 \* a3** | **2** |
| 16 | **a5 + a1 + a2** | **14** |
| 17 | **a5 – a1** | **3** |
| 18 | **a2 + a3 \* a4** | **7** |
| 19 | **0** | **0** |
| 20 | **a3 + a4 \* a5** | **0** |

Chương trình a đọc lần lượt các giá trị input và cho vào stack S1 nếu chúng khác 0. Khi bắt gặp phần tử có giá trị 0 thì pop phần tử trên cùng của Stack ra, sau đấy enqueue lần lượt các giá trị còn lại vào queue Q1 đến khi stack rỗng. Vòng lặp lặp lại cho đến khi đọc hết input. Sau vòng lặp:

S1: rỗng

Q1: 9 -> -5 -> 5 -> 7 -> 9 -> 7 -> 7 -> 5 -> 3 -> 14 -> 2

Chương trình b đọc lần lượt các giá trị input và cho vào Queue Q1 nếu chúng khác 0. Khi bắt gặp phần tử có giá trị 0, chương trình sẽ lấy giá trị mà rear đang chỉ đến và push giá trị đó vào queue thay vì push giá trị 0. Vòng lặp chạy cho đến khi đọc hết input. Sau vòng lặp:

Q1: 9 -> 7 -> 7 -> 7 -> 5 -> 7 -> 7 -> 9 -> 7 ->5 -> -5 ->14 -> 14 ->14 -> 2 -> 14 -> 3 -> 7 -> 7 -> 7

**LAB 4**

**Question 1: Stack**

#include<iostream>

#include<sstream>

#include<stdbool.h>

using namespace std;

struct Node {

int data;

Node \*next;

};

struct Stack {

int counter;

Node \*top;

};

Stack\* createStack(){

Stack \*result;

result = new Stack;

result->counter = 0;

result->top = NULL;

return result;

}

void pushStack(Stack \*&s, int data){

Node \*pTemp;

if (s->counter == 0){

pTemp = new Node;

s->top = pTemp;

pTemp->data = data;

pTemp->next = NULL;

}

else{

pTemp = new Node;

pTemp->data = data;

pTemp->next = s->top;

s->top = pTemp;

}

s->counter++;

}

void popStack(Stack \*&s){

Node \*pTemp = s->top;

s->top = s->top->next;

s->counter--;

delete (pTemp);

}

int topStack(Stack \*s){

if (s->counter == 0){

return 0;

}

else {

return s->top->data;

}

}

bool isEmpty(Stack \*s){

if (s->top == NULL){

return true;

}

else {

return false;

}

}

bool isFull(Stack \*s, int sizeStack){

if (s->counter == sizeStack){

return true;

}

else {

return false;

}

}

void clearStack (Stack \*&s){

while(!isEmpty(s)){

popStack(s);

}

}

int sizeStack(Stack \*s){

return s->counter;

}

void printStack(Stack \*s){

if (s->counter == 0){

cout<<"Stack is empty. ";

}

else{

Node \*pTemp;

pTemp = s->top;

while (pTemp != NULL){

cout<<pTemp->data<<" ";

pTemp = pTemp->next;

}

}

}

int main() {

Stack \*s1;

//1a

s1 = createStack();

//1b

pushStack(s1, 5);

pushStack(s1, 6);

pushStack(s1, 7);

printStack(s1);

cout<<"\n";

//1c

popStack(s1);

printStack(s1);

cout<<"\n";

//1d

int n;

n = topStack(s1);

cout<<n;

cout<<"\n";

//1h

int Size;

Size = sizeStack(s1);

cout<<Size;

cout<<"\n";

//1e

bool check;

check = isEmpty(s1);

cout<<check;

cout<<"\n";

//1f

bool check\_full;

check\_full = isFull(s1, 100);

cout<<check\_full;

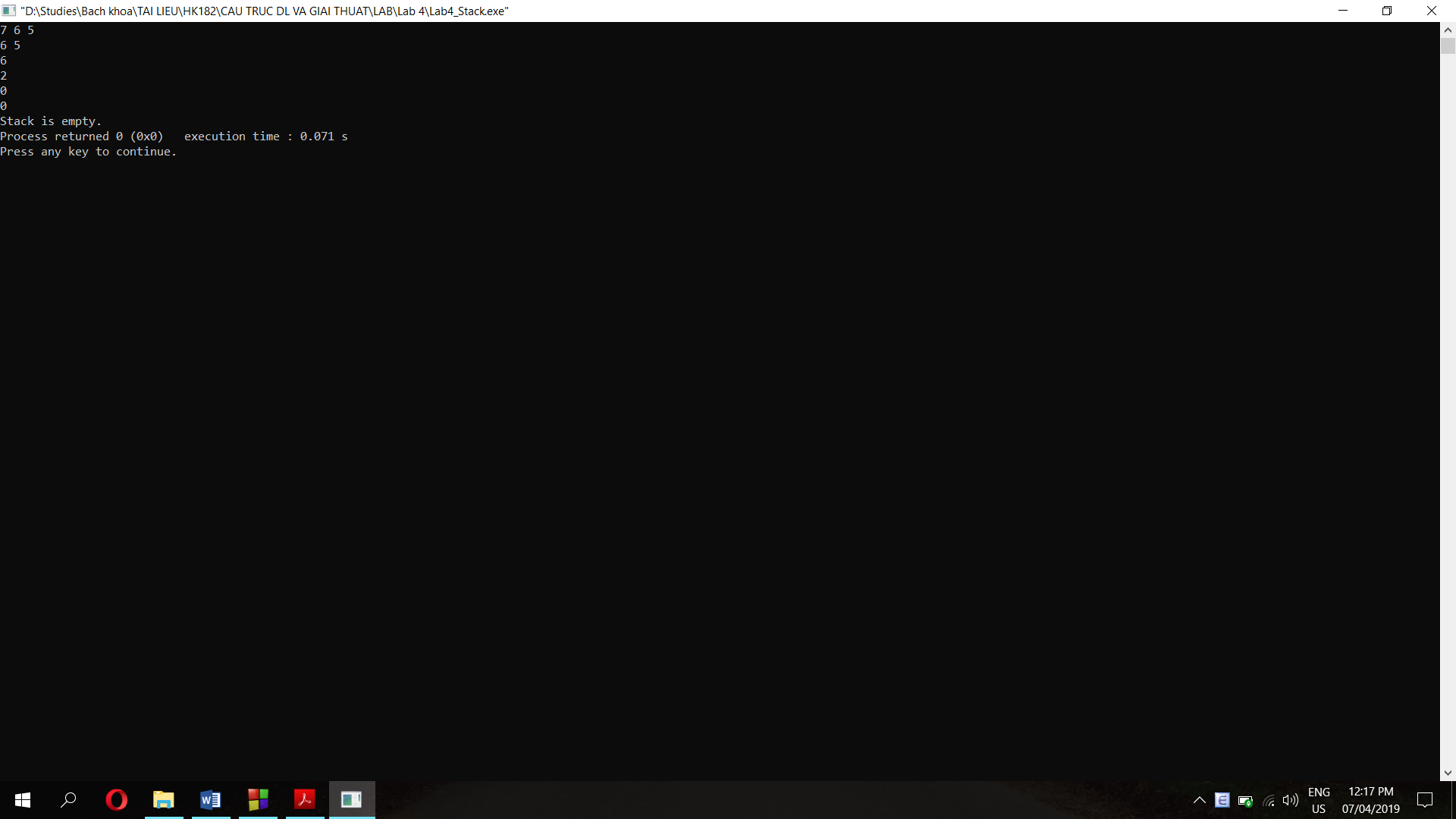
cout<<"\n";

//1g

clearStack(s1);

printStack(s1);

}



**Question 2: Queue**

#include<iostream>

#include<sstream>

#include<stdbool.h>

using namespace std;

struct Node {

int data;

Node \*next;

};

struct Queue {

int counter;

Node \*Front;

Node \*Rear;

};

Queue\* createQueue(){

Queue \*result;

result = new Queue;

result->counter = 0;

result->Front = NULL;

result->Rear = NULL;

return result;

}

void enQueue(Queue \*&q, int data){

if (q->counter == 0){

Node \*pTemp = new Node;

q->Front = pTemp;

q->Rear = pTemp;

pTemp->data = data;

pTemp->next = NULL;

}

else{

Node \*pTemp = new Node;

pTemp->data = data;

q->Rear->next = pTemp;

q->Rear = pTemp;

pTemp->next = NULL;

}

q->counter++;

}

void deQueue(Queue \*&q){

if (q->counter == 0){

return;

}

else if (q->counter == 1){

Node \*pTemp = q->Front;

q->Front = NULL;

q->Rear = NULL;

delete(pTemp);

}

else{

Node \*pTemp = q->Front;

q->Front = pTemp->next;

delete(pTemp);

}

q->counter--;

}

int QueueFront(Queue \*q){

if (q->counter == 0){

return 0;

}

else {

return q->Front->data;

}

}

int QueueRear(Queue \*q){

if (q->counter == 0){

return 0;

}

else {

return q->Rear->data;

}

}

bool isEmpty(Queue \*q){

if (q->Front == NULL){

return true;

}

else {

return false;

}

}

bool isFull(Queue \*q, int sizeQueue){

if (q->counter >= sizeQueue){

return true;

}

else {

return false;

}

}

void clearQueue(Queue \*&q){

while(!isEmpty(q)){

deQueue(q);

}

}

int sizeQueue(Queue \*q){

return q->counter;

}

void printQueue(Queue \*q){

if (q->counter == 0){

cout<<"Queue is empty. ";

}

else{

Node \*pTemp;

pTemp = q->Front;

while (pTemp != NULL){

cout<<pTemp->data<<" ";

pTemp = pTemp->next;

}

}

}

int main() {

Queue \*Q;

//1a

Q = createQueue();

//1b

enQueue(Q, 1);

enQueue(Q, 2);

enQueue(Q, 3);

printQueue(Q);

cout<<"\n";

//1c

deQueue(Q);

printQueue(Q);

cout<<"\n";

//1d

cout<<QueueFront(Q);

cout<<"\n";

//1e

cout<<QueueRear(Q);

cout<<"\n";

//1f

cout<<isEmpty(Q);

cout<<"\n";

//1g

cout<<isFull(Q, 100);

cout<<"\n";

//1i

cout<<sizeQueue(Q);

cout<<"\n";

//1h

clearQueue(Q);

printQueue(Q);

cout<<"\n";

}

