CSE2003 Data Structures and Algorithms

**Assessment 1**

**Rishav Raj Jain**

**17BCE0042**

**L29-L30**

1. **Linear Search and Binary Search**

**CODE**

#include<stdio.h>

#include<stdlib.h>

void creation();

void display();

void linear();

void binary();

void dup();

void occ();

int a[10],i=0,e,n,t,top=-1,s,r=5;

main()

{

printf(" 17BCE0042\n");

char option='T';

while(option=='T')

{

printf("\n");

printf("Enter command (17BCE0042)\n");

printf("1. Creation\n");

printf("2. Display\n");

printf("3. Binary search\n");

printf("4. Linear Search\n");

printf("5. Find Duplicates\n");

printf("6. Find Occurrence\n");

printf("7. Exit\n");

scanf("%d",&n);

switch(n)

{

case 1:

creation();

break;

case 2:

display();

break;

case 3:

binary();

break;

case 4:

linear();

break;

case 5:

dup();

break;

case 6:

occ();

break;

case 7:

printf(" Thankyou!!\n");

exit(0);

}

}

}

void creation()

{

printf("Enter your element\n");

for (i=0;i<r;i++)

{

scanf("%d",&e);

a[i]=e;

}

}

void display()

{

printf("\n");

for (i=0;i<r;i++)

{

printf("%d\n",a[i]);

}

}

void linear()

{

printf("Enter your element\n");

scanf("%d",&s);

int coun=0;

for (i=0;i<r;i++)

{

if(a[i]==s)

{

coun++;

break;

}

}

if (coun>0)

{

printf("Element found\n");

}

else{

printf("Element Not found\n");

}

}

void binary()

{

int f=0,l=r-1,m= (f+l)/2;

printf("Enter your element\n");

scanf("%d",&s);

int coun=0;

while(f<=l)

{

m= (f+l)/2;

if (s==a[m])

{

coun++;

break;

}

else if(s>a[m])

{

f=m+1;

}

else if(s<a[m])

{

l=m-1;

}

}

if (coun>0)

{

printf("Element found\n");

}

else{

printf("Element Not found\n");

}

}

void dup()

{

printf("Enter your element\n");

scanf("%d",&s);

int coun=0;

for (i=0;i<r;i++)

{

if(a[i]==s)

{

coun++;

}

}

if (coun>1)

{

printf("Duplicates found %d\n",coun-1);

}

else{

printf("Duplicates Not found\n");

}

}

void occ()

{

printf("Enter your element\n");

int lo=-1,b[10];

scanf("%d",&s);

int coun=0;

for (i=0;i<r;i++)

{

if(a[i]==s)

{

lo++;

b[lo]=i;

}

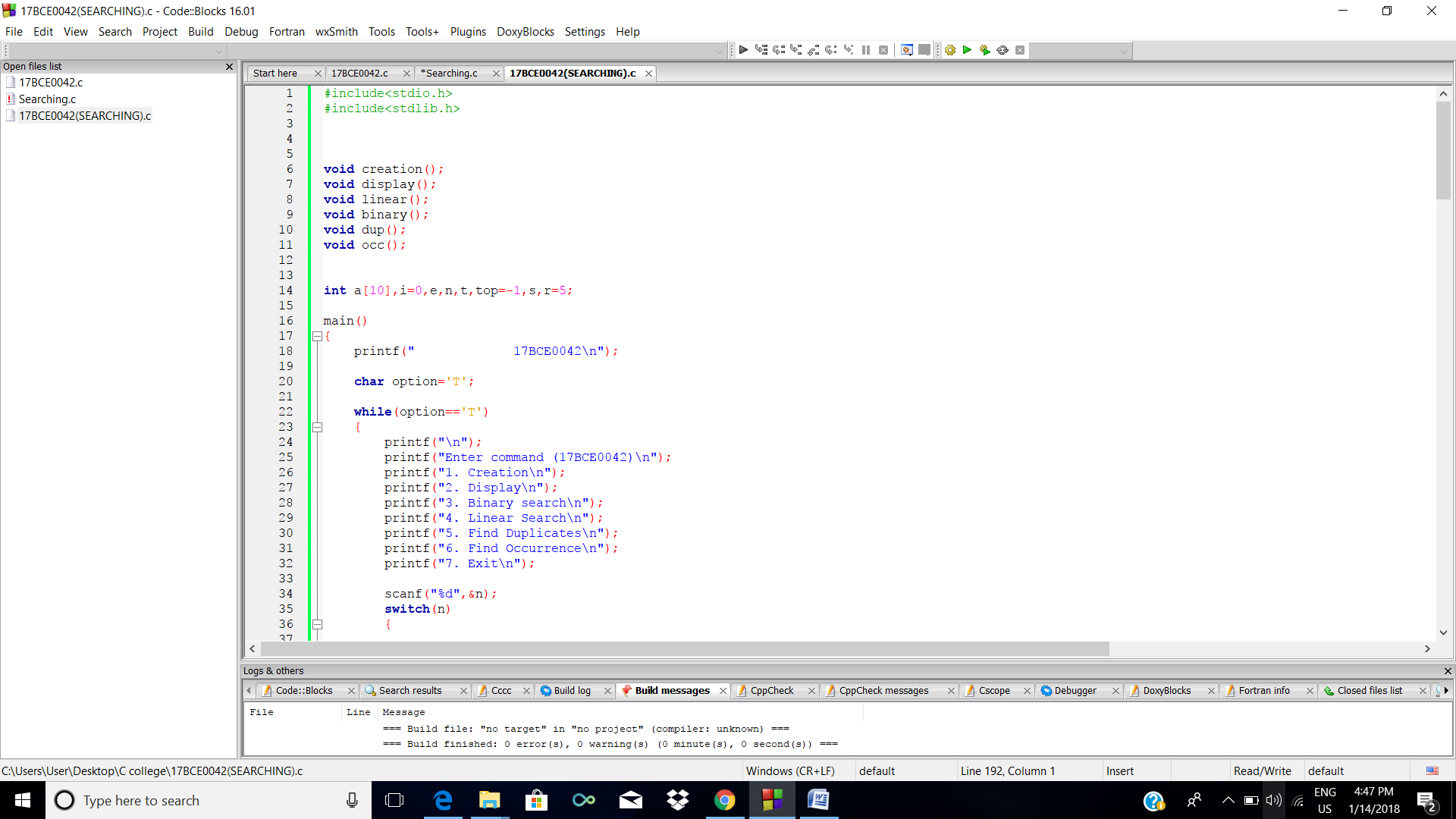
}

printf("First occurrence %d\n",b[0]+1);

printf("Last occurrence %d\n",b[lo]+1);

}

**SCREENSHOTS**



**Taking elements and Displaying**

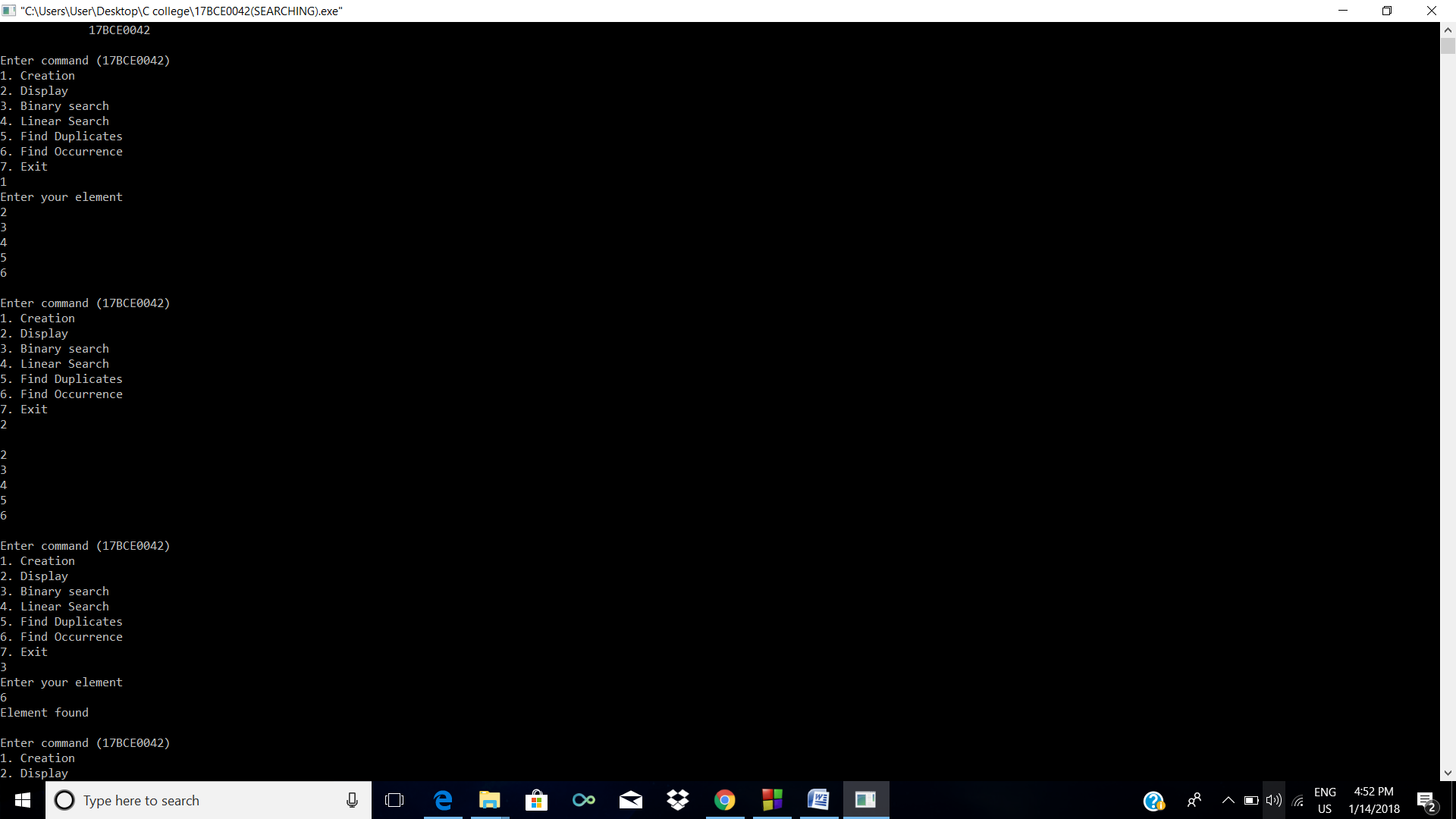
2

3

4

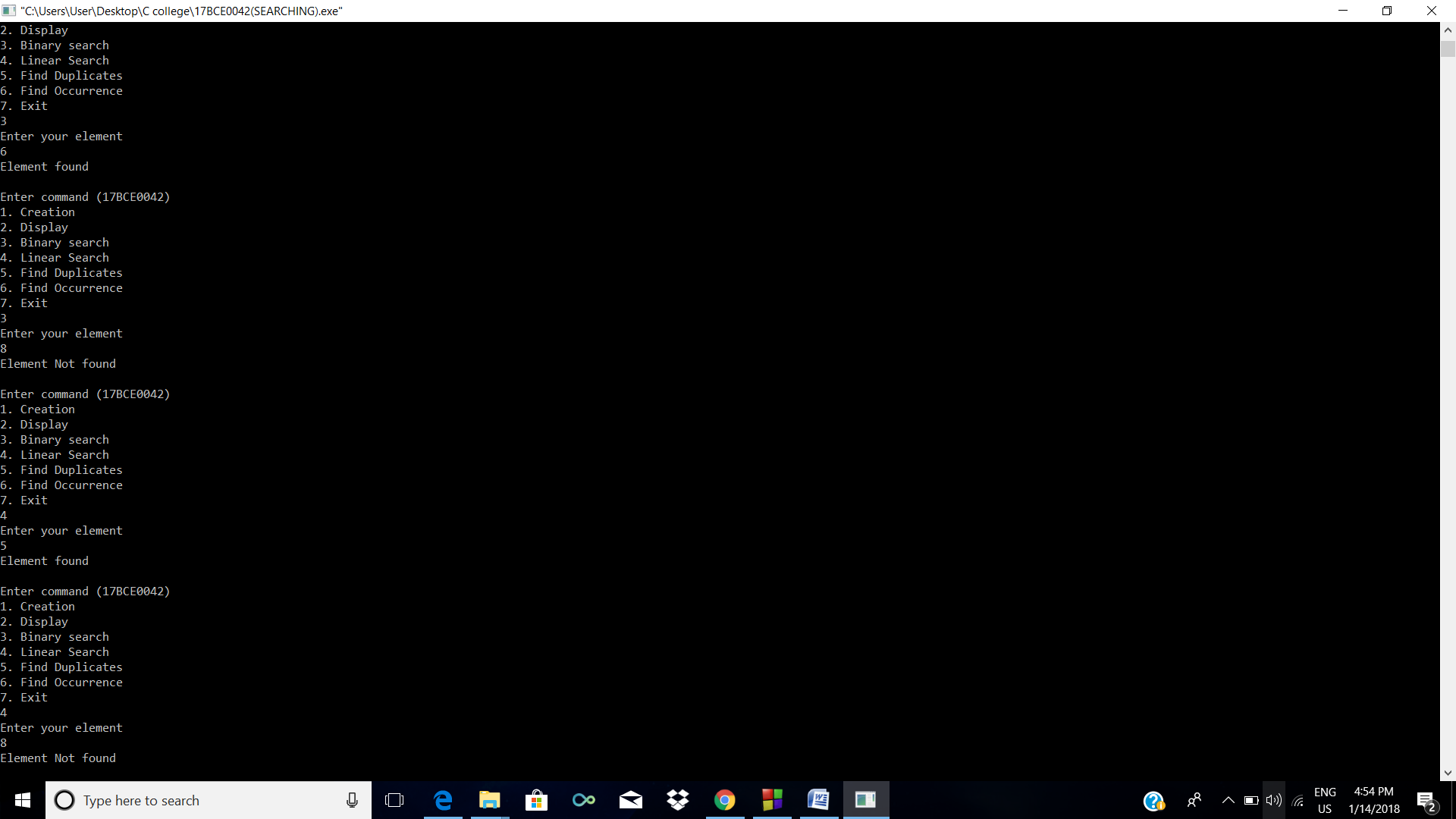
5

6

****

**Binary Search and Linear Search**: **Case 1** Element found

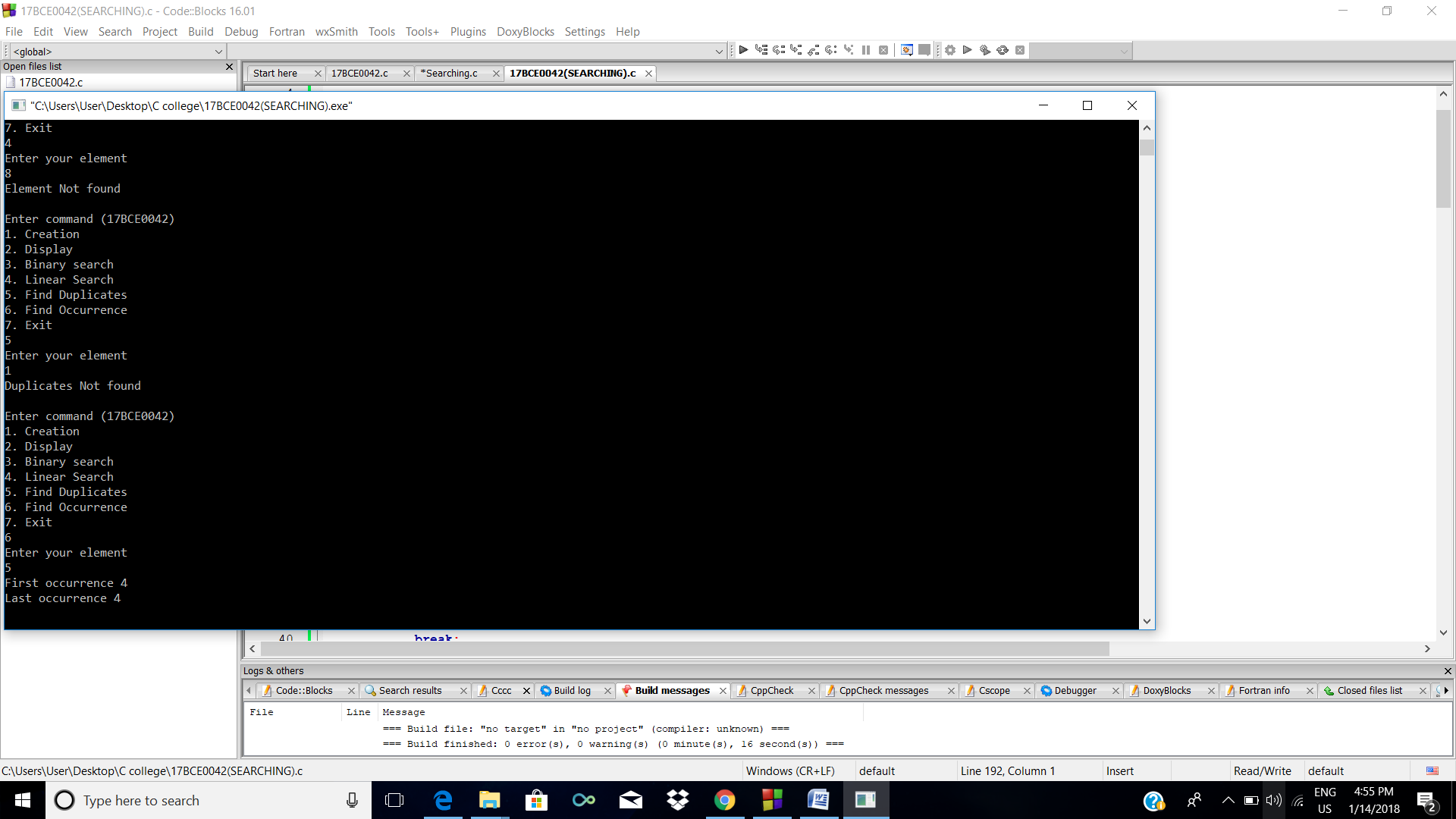
**Case 2** Element not found

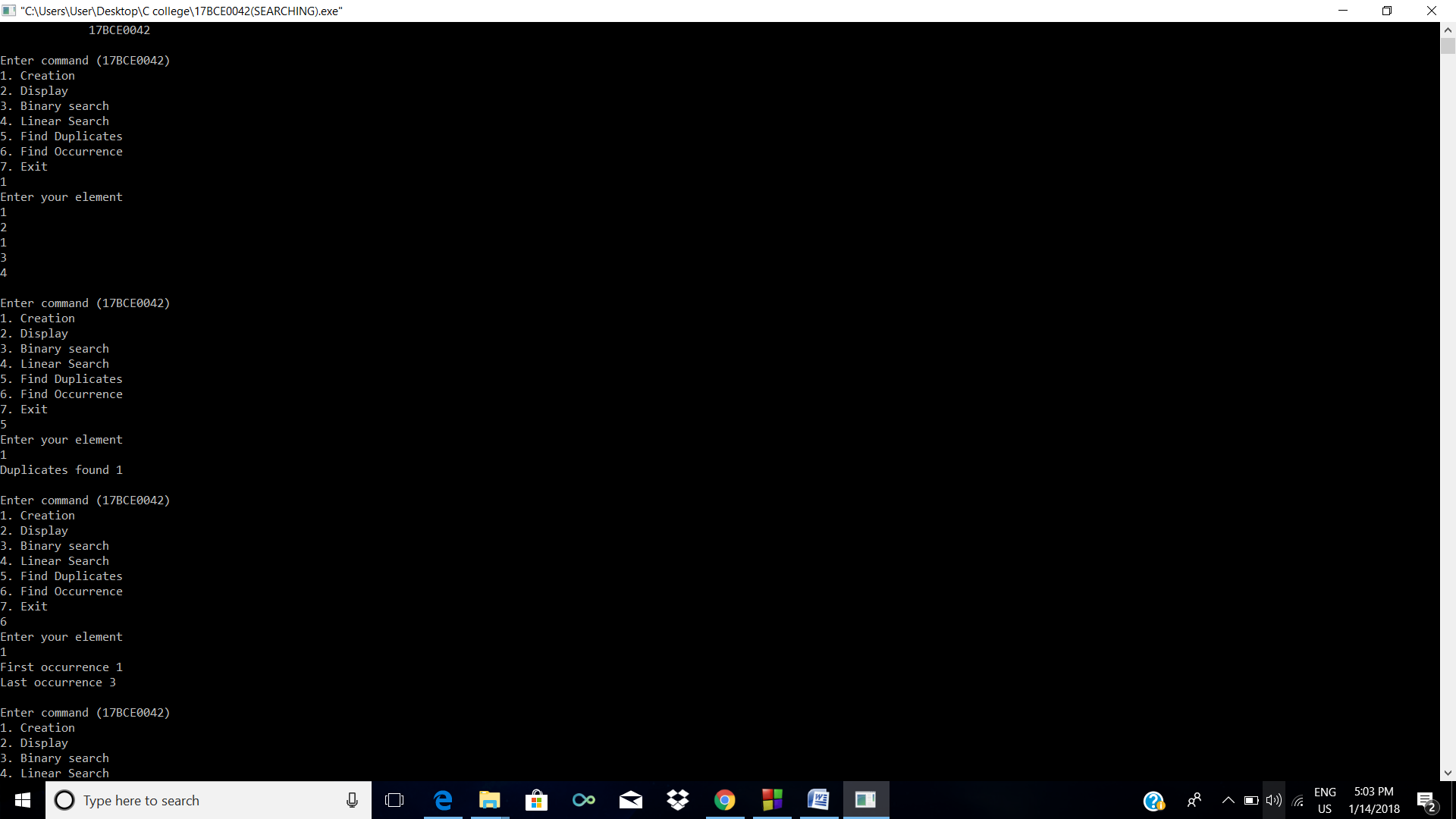
****

**Finding Duplicates and Occurrence**

**Case 1:** No duplicate and occurrence

**Case 2:** With duplicate and occurrence

****

****

**2.Arrays and Stack Interfaces**

**a) Array**

**CODE**

#include<stdio.h>

#include<stdlib.h>

void creation();

void delet();

void display();

void insert();

void update();

int a[10],i=0,e,n,t,top=-1;

main()

{

printf(" 17BCE0042\n");

char option='T';

while(option=='T')

{

printf("\n");

printf("Enter command (17BCE0042)\n");

printf("1. Creation\n");

printf("2. Delete\n");

printf("3. Display\n");

printf("4. Insert\n");

printf("5. Update\n");

printf("6. Exit\n");

scanf("%d",&n);

switch(n)

{

case 1:

creation();

break;

case 2:

delet();

display();

break;

case 3:

display();

break;

case 4:

insert();

display();

break;

case 5:

update();

display();

break;

case 6:

printf(" Thankyou!!\n");

exit(0);

}

}

}

void creation()

{

printf("Enter your element\n");

for (i=0;i<5;i++)

{

top++;

scanf("%d",&e);

a[i]=e;

}

}

void delet()

{

printf("Enter your position\n");

scanf("%d",&t);

for (i=t-1;i<top;i++)

{

a[i]=a[i+1];

}

top--;

}

void insert()

{

printf("Enter your position\n");

scanf("%d",&t);

printf("Enter number\n");

scanf("%d",&e);

for (i=top;i>t-2;i--)

{

a[i+1]=a[i];

}

top++;

a[t-1]=e;

}

void update()

{

printf("Enter your position\n");

scanf("%d",&t);

printf("Enter number\n");

scanf("%d",&e);

a[t-1]=e;

}

void display()

{

printf("\nStatus\n");

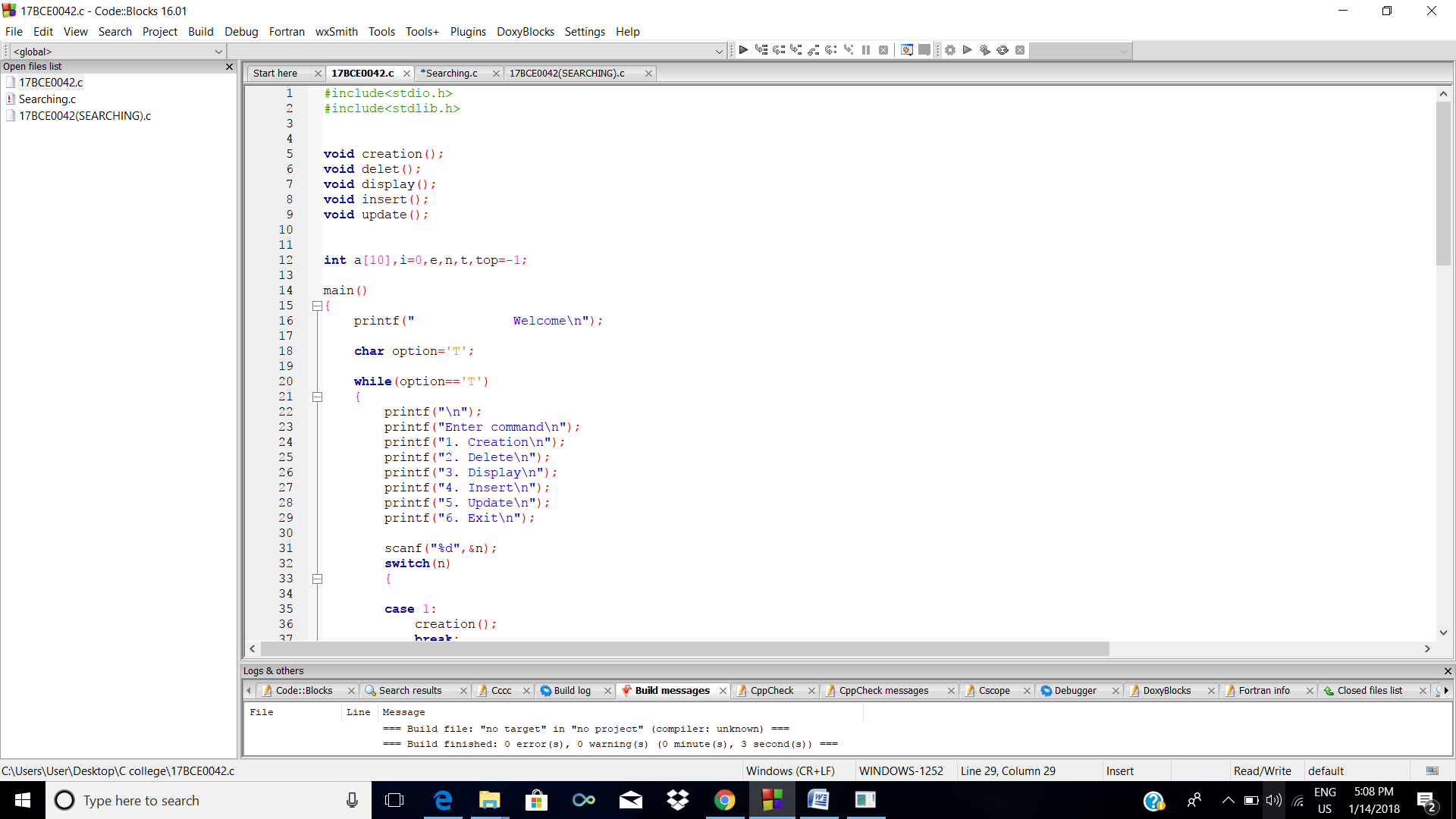
for (i=0;i<=top;i++)

{

printf("%d\n",a[i]);

}

}

****

**Taking elements and Displaying**

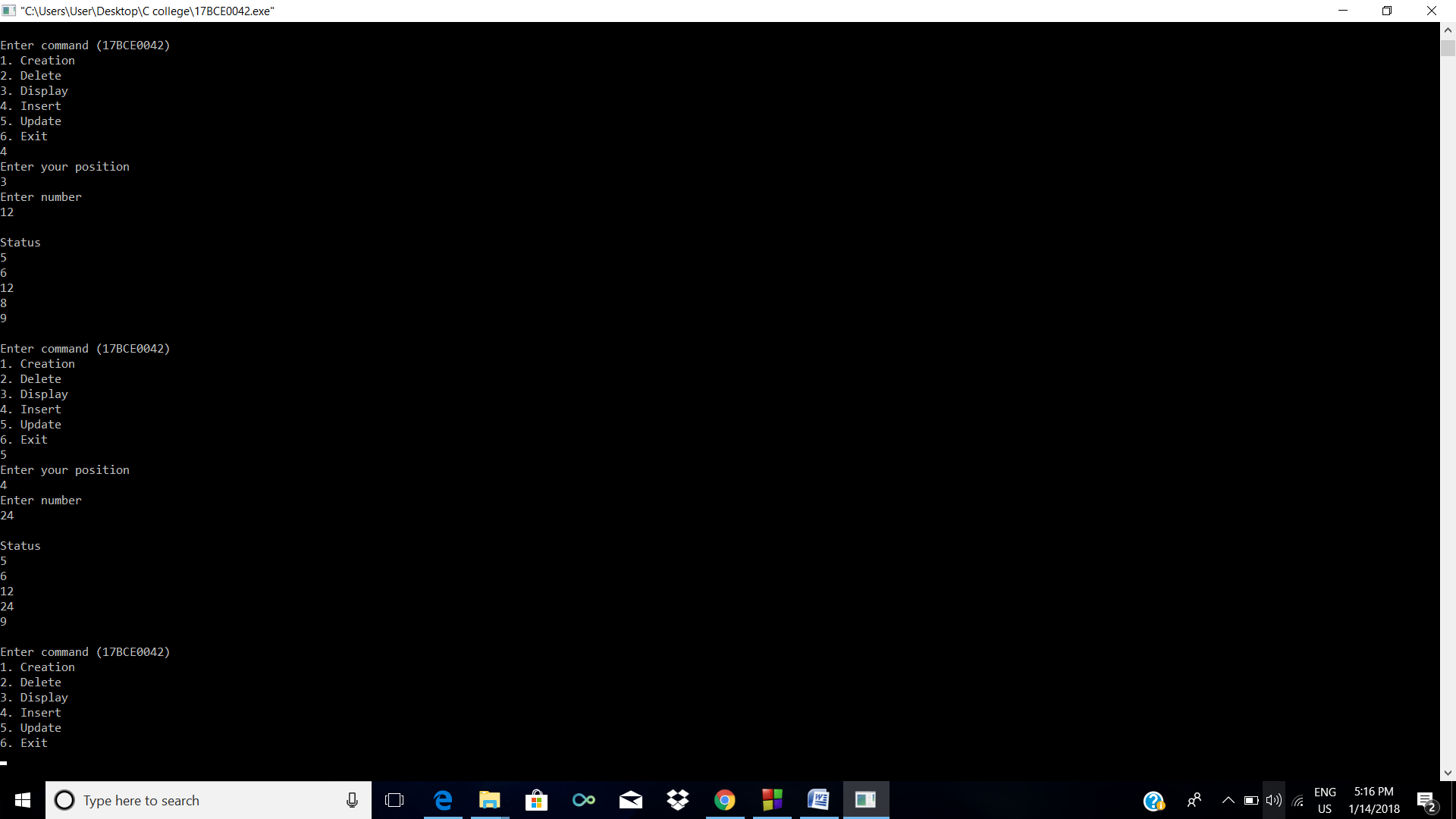
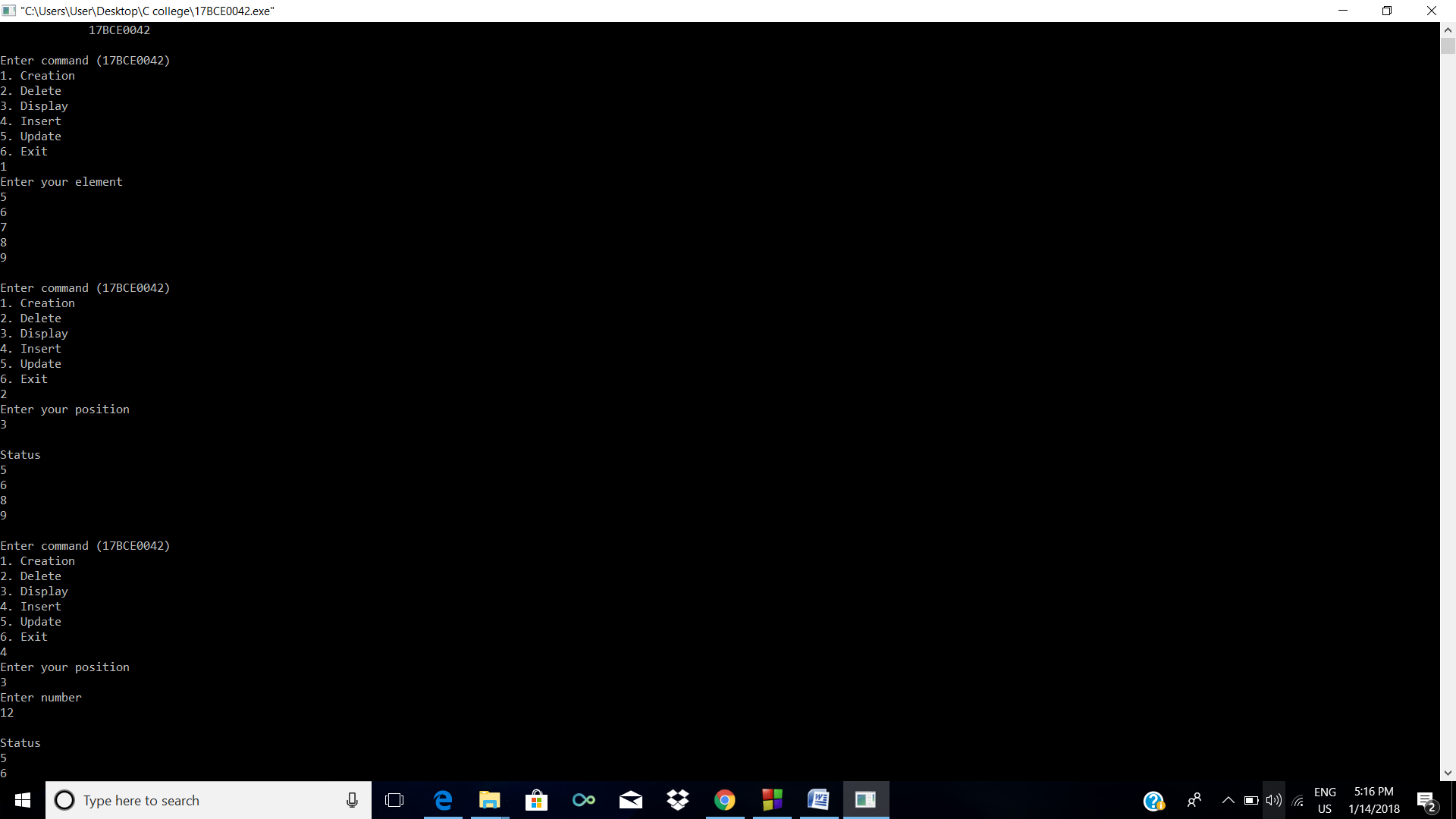
2

3

4

5

6



**b) Stack**

**CODE**

#include<stdio.h>

#include<stdlib.h>

void creation(void);

void push(void);

void display(void);

void pop(void);

void isEmpty(void);

void isFull(void);

int a[10],i=0,e,n,t,top=-1;

main()

{

printf(" 17BCE0042\n");

char option='T';

while(option=='T')

{

printf("\n");

printf("Enter command (17BCE0042)\n");

printf("1. Creation\n");

printf("2. Push\n");

printf("3. Pop\n");

printf("4. Display\n");

printf("5. isEmpty\n");

printf("6. isFull\n");

printf("7. Peek\n");

printf("8. Exit\n\n");

scanf("%d",&n);

switch(n)

{

case 1:

creation();

break;

case 2:

push();

display();

break;

case 3:

pop();

display();

break;

case 4:

display();

break;

case 5:

isEmpty();

break;

case 6:

isFull();

break;

case 7:

printf("\nPeek is %d\n",top+1);

break;

case 8:

printf(" Thankyou!!\n");

exit(0);

}

}

}

void creation()

{

printf("Enter your element\n");

for (i=0;i<5;i++)

{

top++;

scanf("%d",&e);

a[i]=e;

}

}

void pop()

{

if (top==-1)

{

printf("Sorry\nStack is Empty\n");

return;

}

printf("Done\n");

top--;

}

void push()

{

if (top==4)

{

printf("Sorry\nStack is already full\n");

return;

}

printf("Enter number\n");

scanf("%d",&e);

a[top+1]=e;

top++;

}

void display()

{

printf("\nStatus\n");

for (i=0;i<=top;i++)

{

printf("%d\n",a[i]);

}

}

void isEmpty()

{

if (top==-1)

{

printf("Yes, it is empty");

}

else{

printf("No, it is not empty"); }

}

void isFull()

{

if (top==4)

{

printf("Stack is FULL!!");

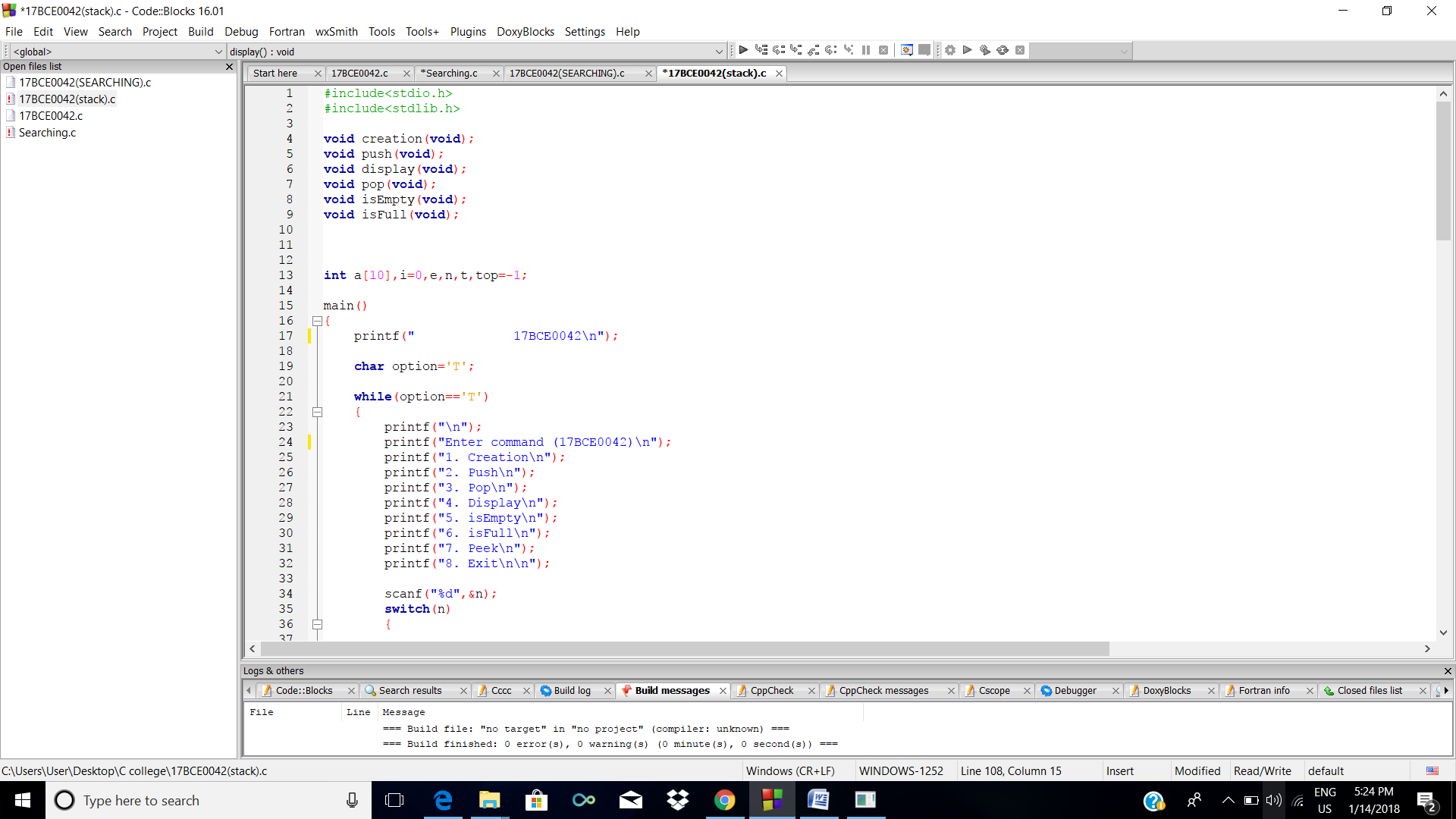
}

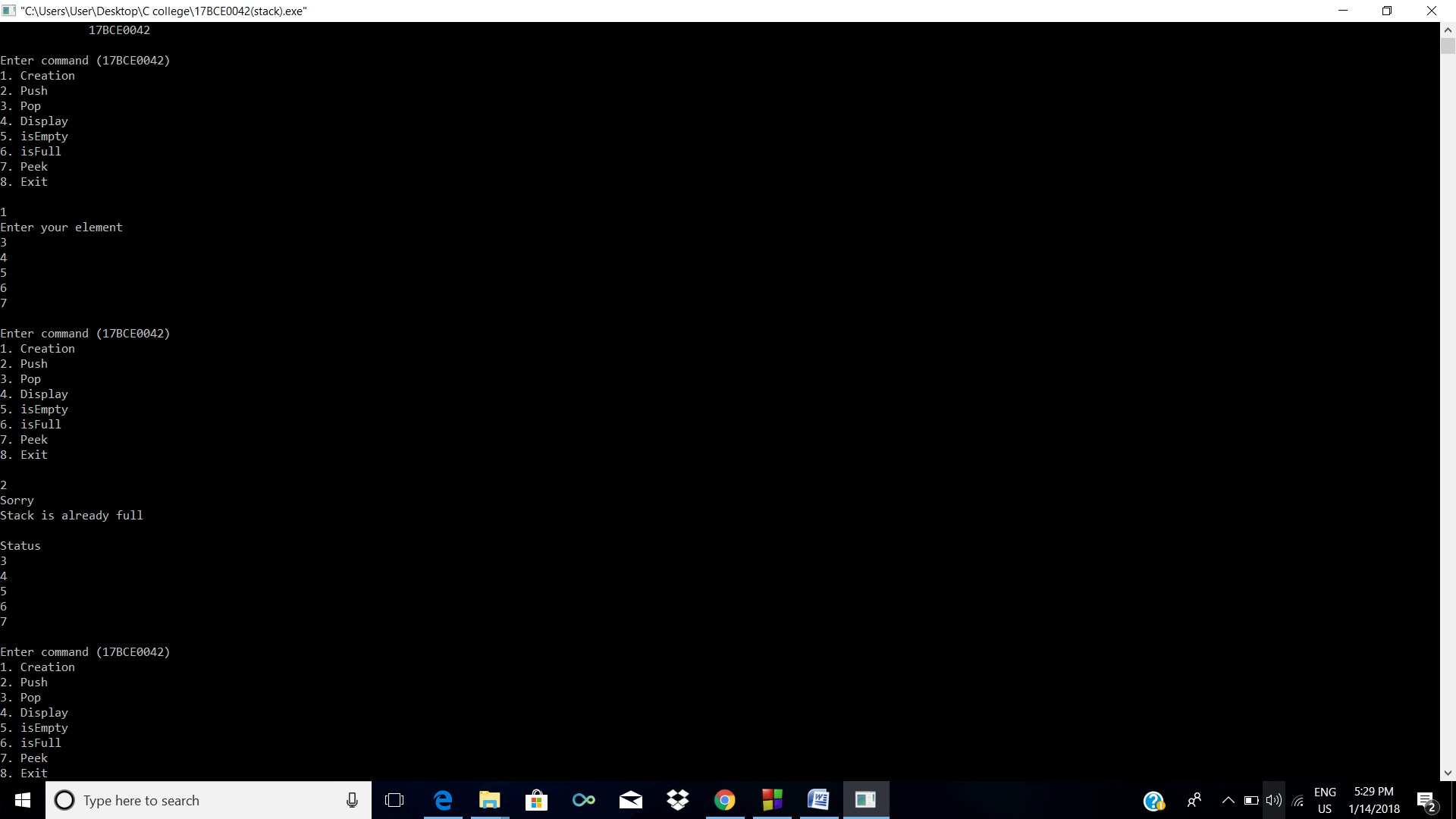
else{

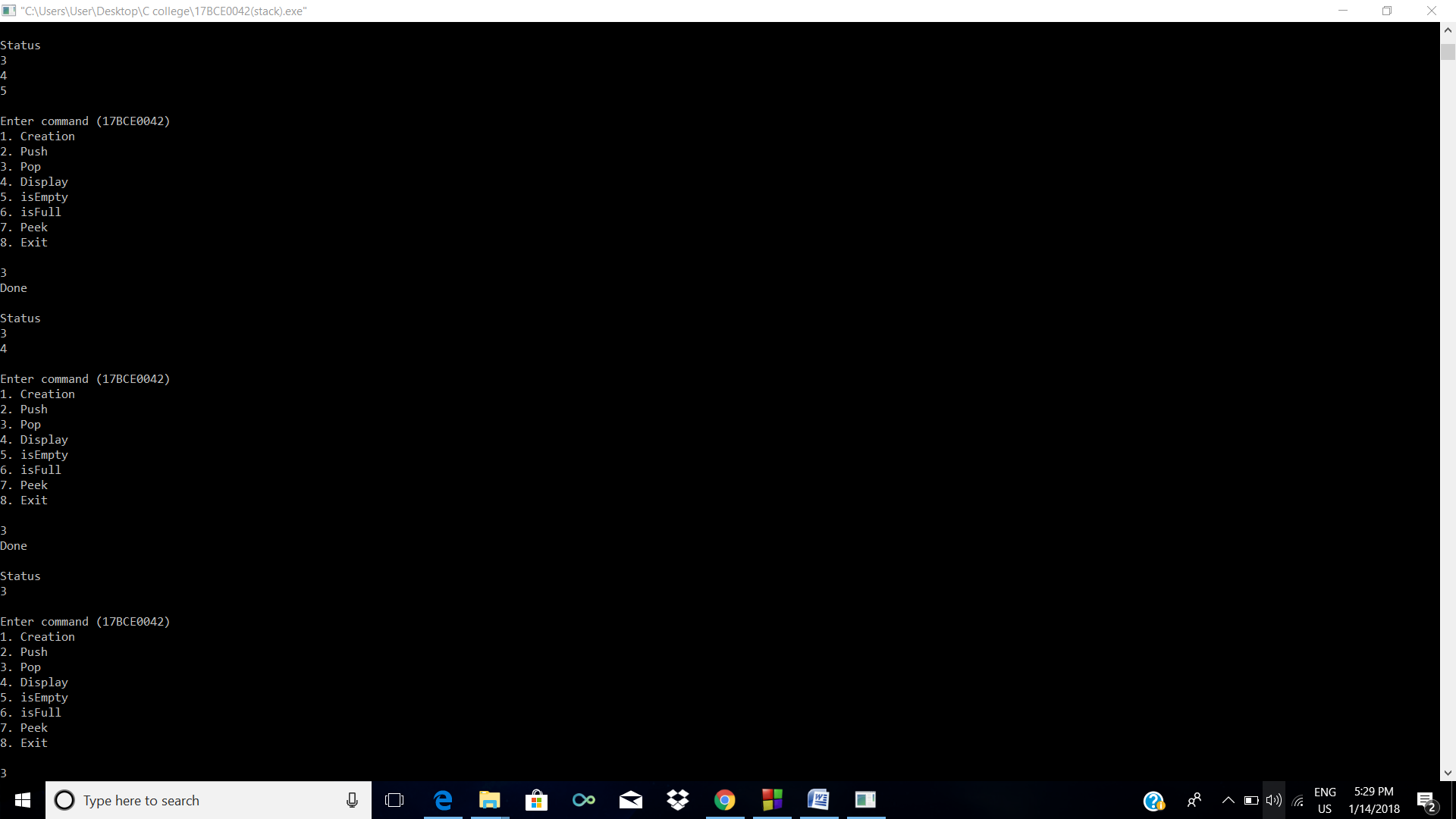
printf("No, it is not Full");

}

}







**3.Applications of Stack**

1. **Infix to Postfix**

**CODE**

#include<stdio.h>

char stack[20];

int top = -1;

void push(char x)

{

stack[++top] = x;

}

char pop()

{

if(top == -1)

return -1;

else

return stack[top--];

}

int priority(char x)

{

if(x == '(')

return 0;

if(x == '+' || x == '-')

return 1;

if(x == '\*' || x == '/')

return 2;

}

main()

{

char exp[20];

char \*temp, x;

printf("Enter the expression :: ");

scanf("%s",exp);

temp = exp;

while(\*temp != '\0')

{

if(isalnum(\*temp))

printf("%c",\*temp);

else if(\*temp == '(')

push(\*temp);

else if(\*temp == ')')

{

while((x = pop()) != '(')

printf("%c", x);

}

else

{

while(priority(stack[top]) >= priority(\*temp))

printf("%c",pop());

push(\*temp);

}

temp++;

}

while(top != -1)

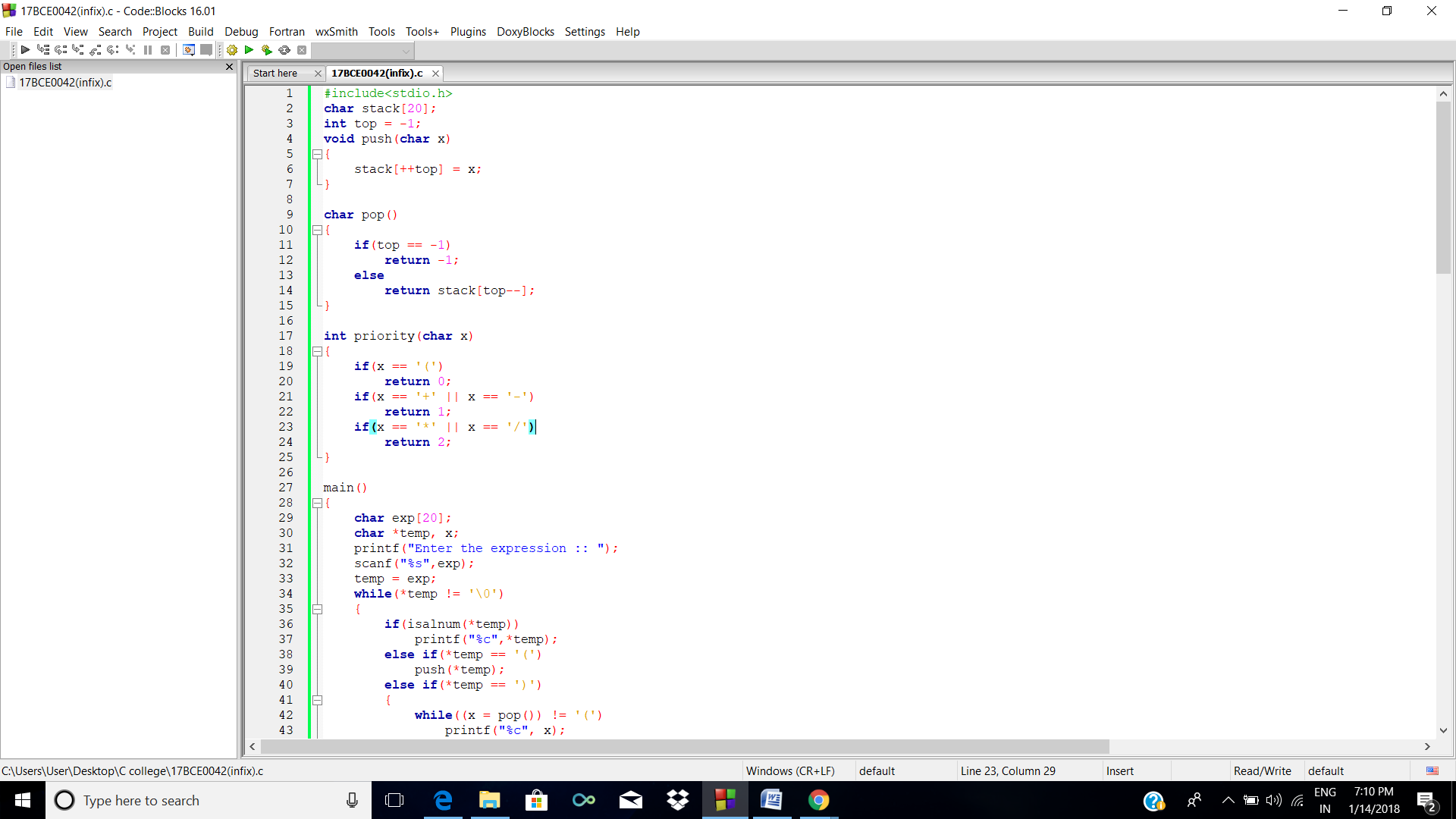
{

printf("%c",pop());

}

}

**SCREENSHOT**



**b. Evaluation of Postfix**

**CODE**

b)

#include<stdio.h>

#include<ctype.h>

#define MAXSTACK 100

#define POSTFIXSIZE 100

int stack[MAXSTACK];

int top = -1 ;

void push(int item)

{

if(top >= MAXSTACK -1)

{

printf("stack over flow");

return;

}

else

{

top = top + 1 ;

stack[top]= item;

}

}

int pop()

{

int item;

if(top <0)

{

printf("stack under flow");

}

else

{

item = stack[top];

top = top - 1;

return item;

}

}

void EvalPostfix(char postfix[])

{

int i ;

char ch;

int val;

int A, B ;

for (i = 0 ; postfix[i] != ')'; i++)

{

ch = postfix[i];

if (isdigit(ch))

{

push(ch - '0');

}

else if (ch == '+' || ch == '-' || ch == '\*' || ch == '/')

{

A = pop();

B = pop();

switch (ch)

{

case '\*':

val = B \* A;

break;

case '/':

val = B / A;

break;

case '+':

val = B + A;

break;

case '-':

val = B - A;

break;

}

push(val);

}

}

printf( " \n Result of expression evaluation : %d \n", pop()) ;

}

int main()

{

int i ;

char postfix[POSTFIXSIZE];

printf( " \nEnter postfix expression,\npress right parenthesis ')' for end expression : ");

for (i = 0 ; i <= POSTFIXSIZE - 1 ; i++)

{

scanf("%c", &postfix[i]);

if ( postfix[i] == ')' )

{

break;

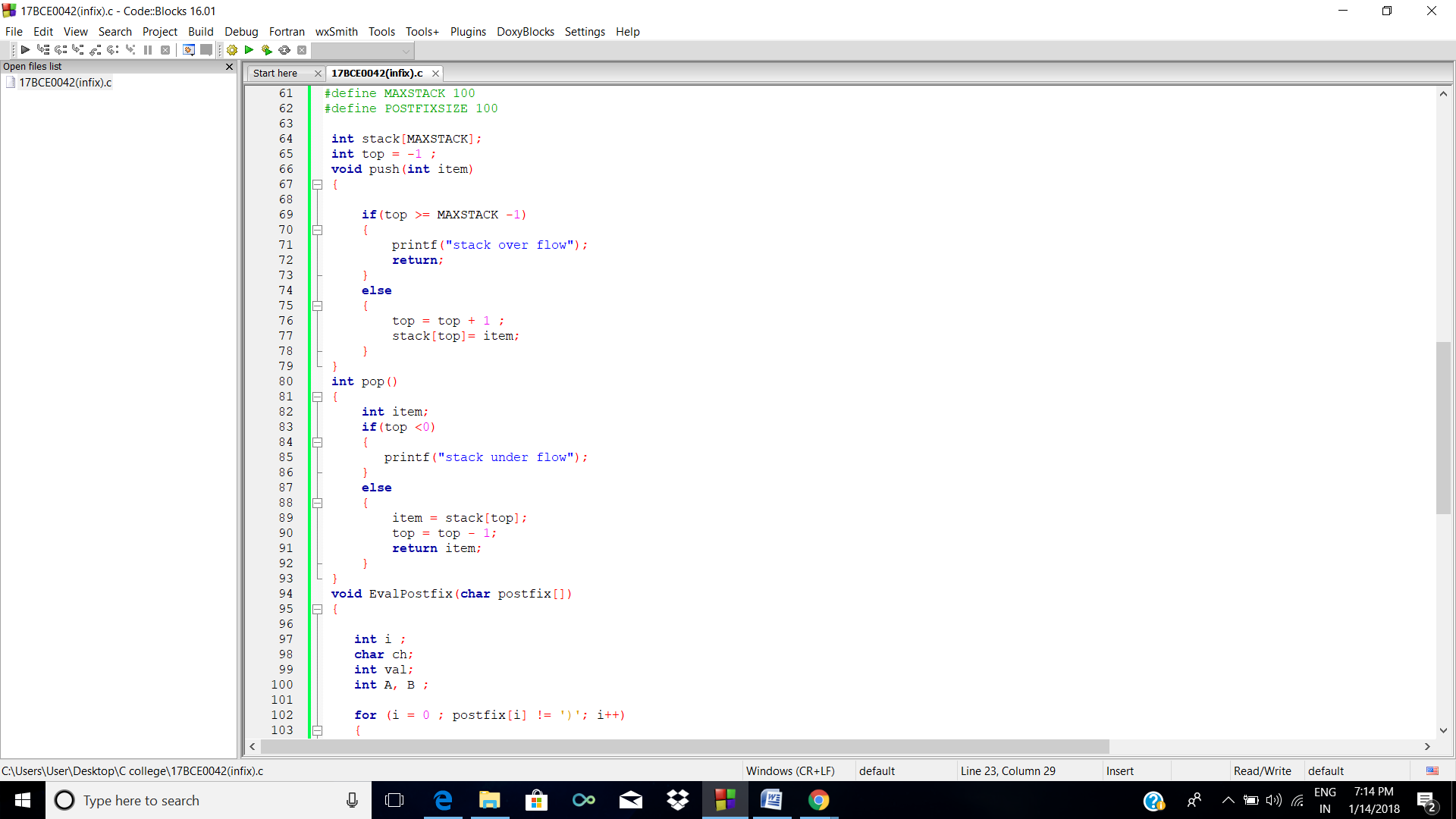
}

}

EvalPostfix(postfix);

return 0;

}



**3.Queue**

**CODE**

#include<stdio.h>

#include<stdlib.h>

void enqueue();

void display();

void dequeue();

void isEmpty();

void isFull();

int a[10],i=0,e,n,t,front=-1,rear=-1;

main()

{

printf(" 17BCE0042\n");

char option='T';

while(option=='T')

{

printf("\n");

printf("Enter command (17BCE0042)\n");

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Display\n");

printf("4. isEmpty\n");

printf("5. isFull\n");

printf("6. Peek\n");

printf("7. Exit\n\n");

scanf("%d",&n);

switch(n)

{

case 1:

enqueue();

display();

break;

case 2:

dequeue();

display();

break;

case 3:

display();

break;

case 4:

isEmpty();

break;

case 5:

isFull();

display();

break;

case 6:

printf("\nPeek is %d\n",a[rear]);

break;

case 7:

printf(" Thankyou!!\n");

exit(0);

}

}

}

void dequeue()

{

if (rear==-1)

{

printf("Sorry\nQueue is Empty\n");

return;

}

for(i=front;i<rear;i++)

{

a[i]=a[i+1];

}

printf("Done\n");

rear--;

}

void enqueue()

{

if (rear==4)

{

printf("Sorry\nQueue is already full\n");

return;

}

if(front==-1)

{

front++;

}

printf("Enter number\n");

scanf("%d",&e);

a[rear+1]=e;

rear++;

}

void display()

{

printf("\n");

for (i=0;i<=rear;i++)

{

printf("%d\n",a[i]);

}

}

void isEmpty()

{

if (rear==-1)

{

printf("Yes, it is empty");

}

else

{

printf("No it is not empty");

}

}

void isFull()

{

if (rear==4)

{

printf("Queue is FULL!!");

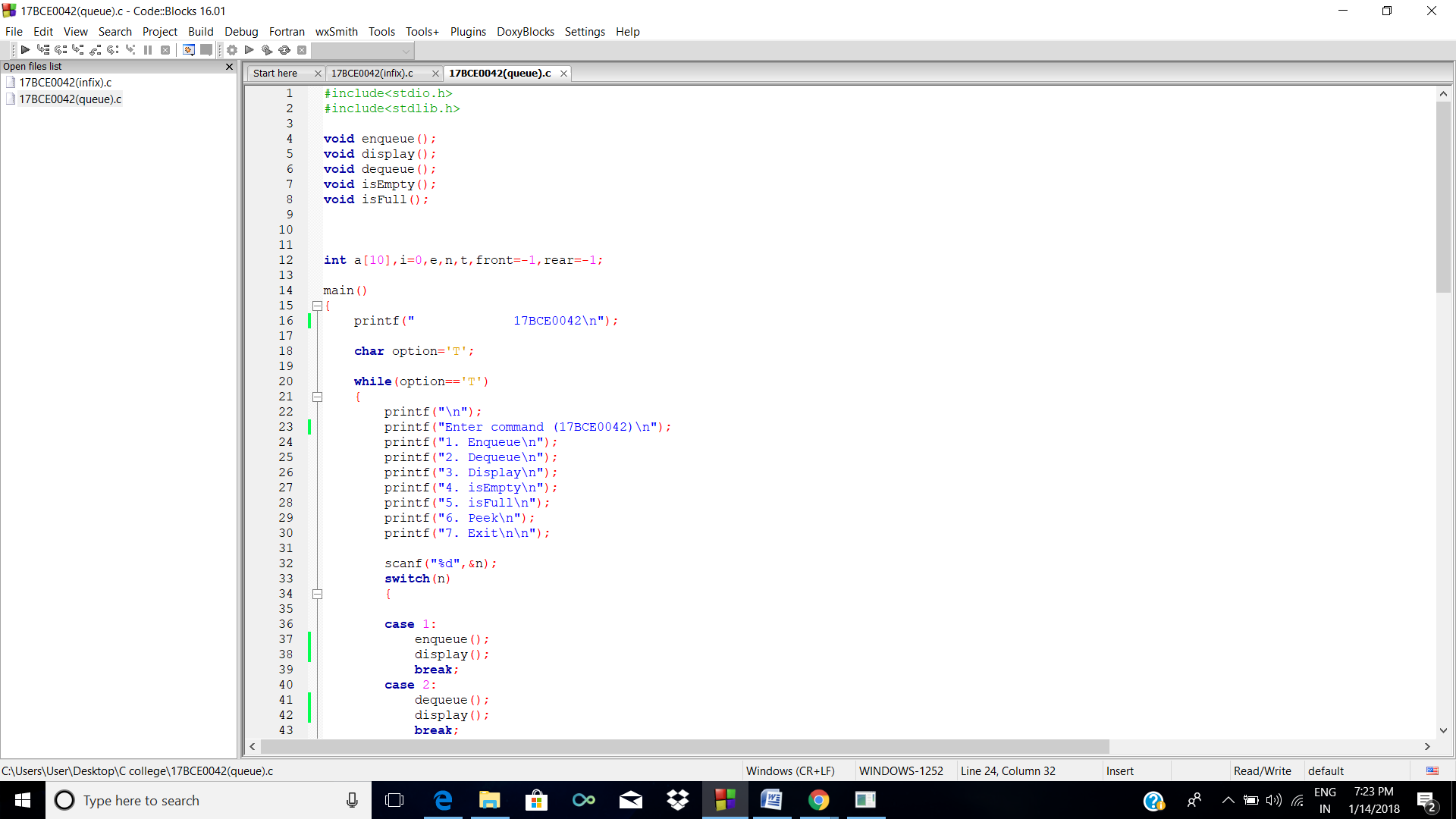
}

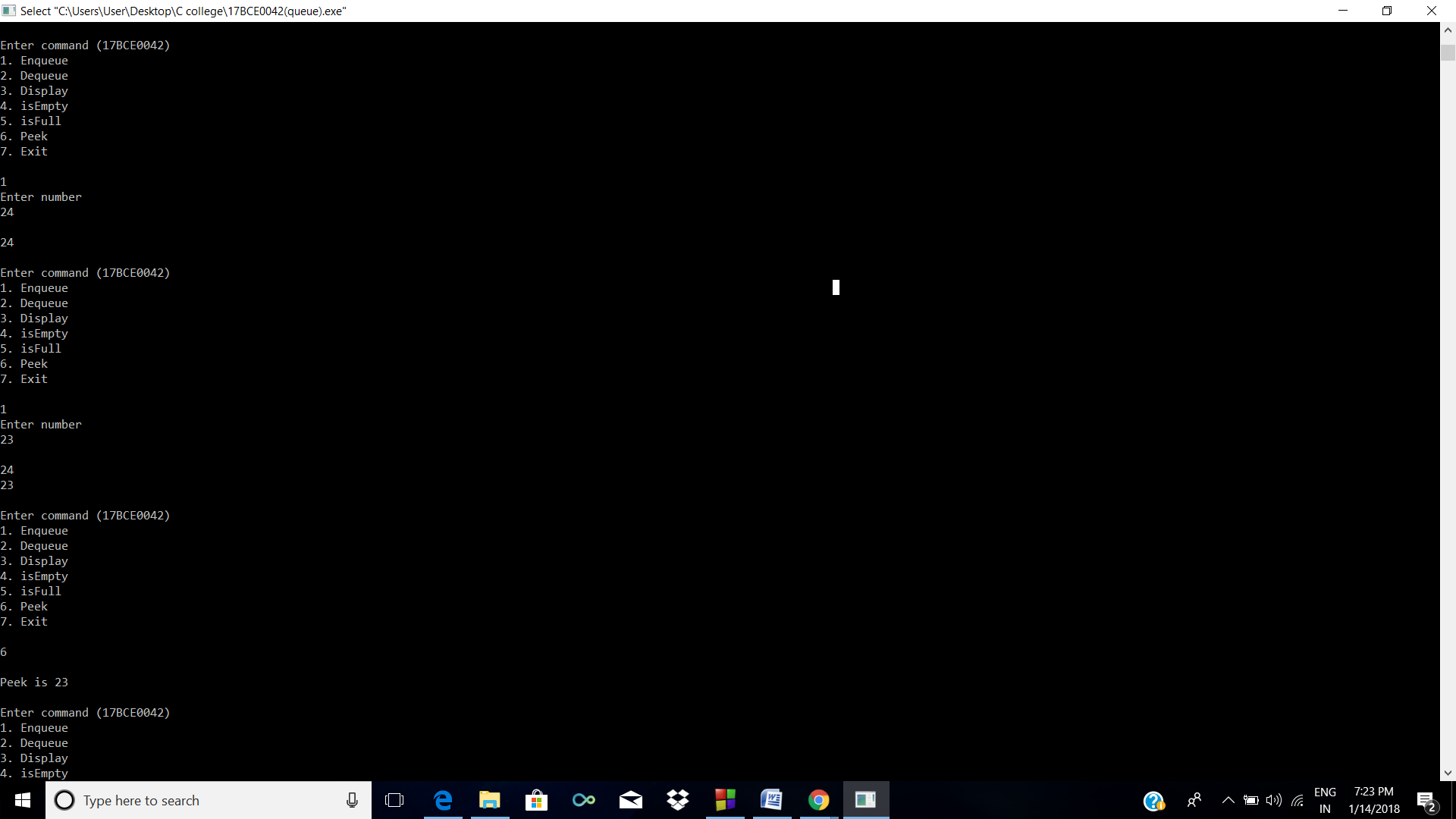
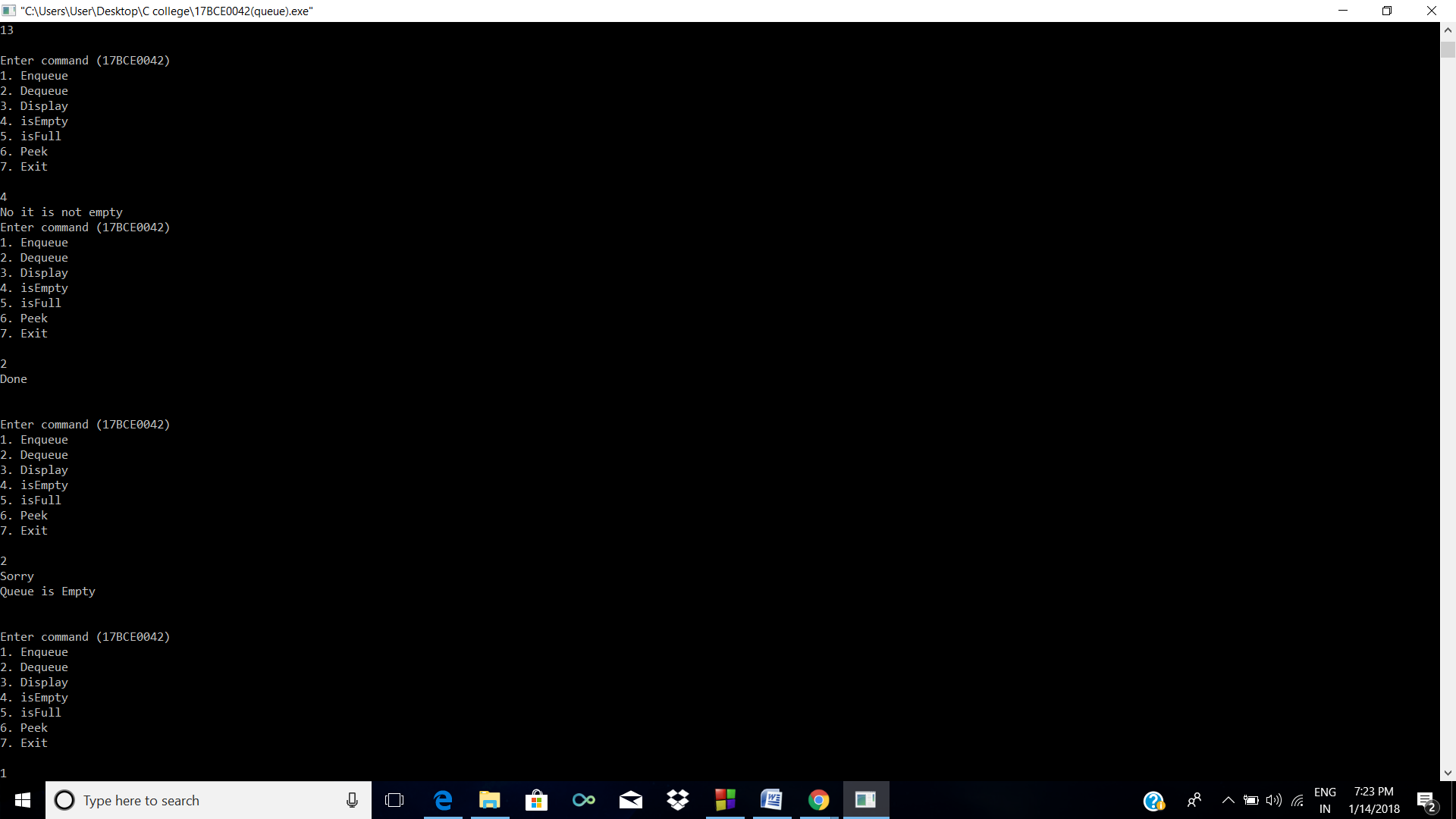
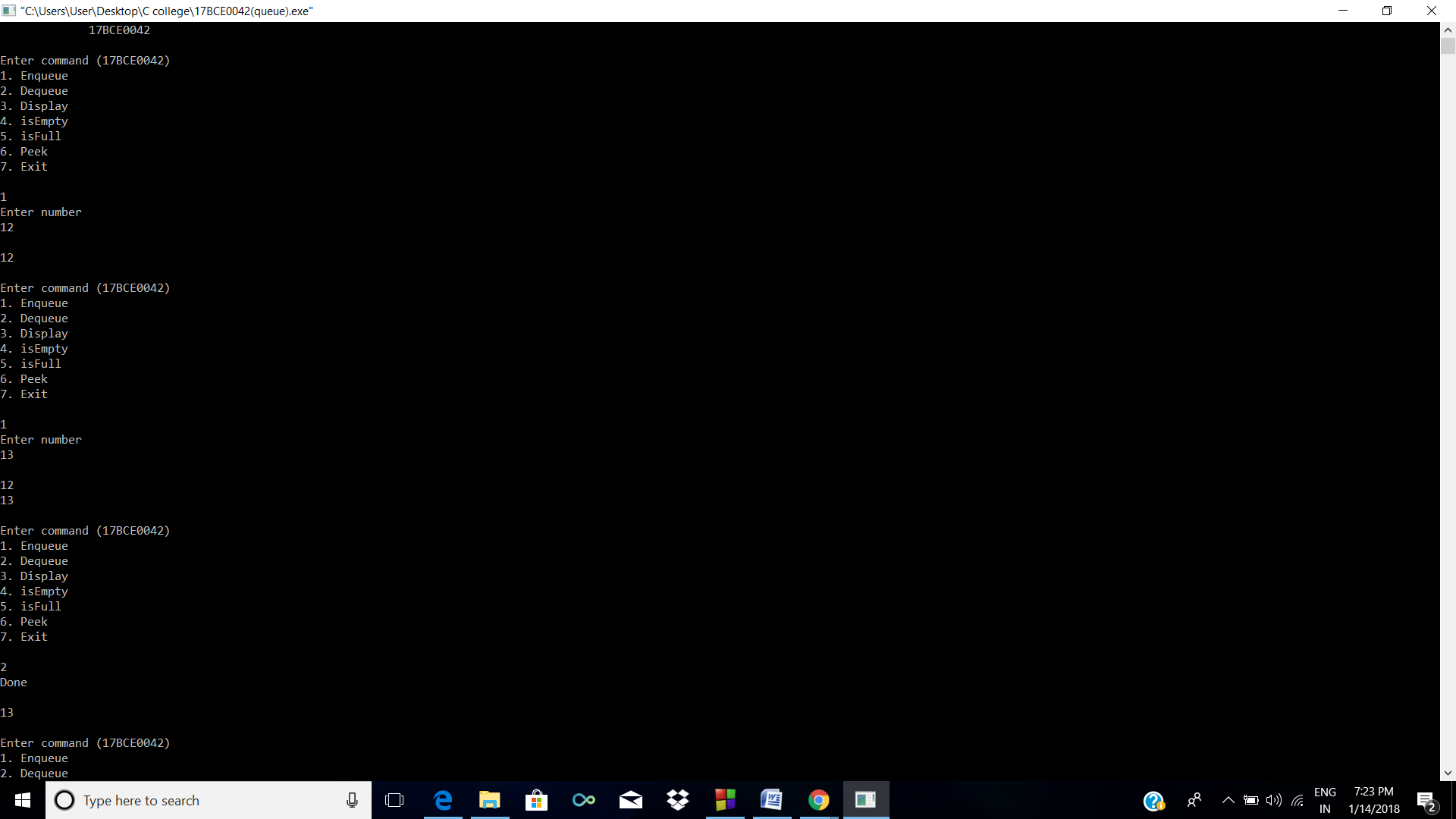
else{

printf("No it is not full");

}

}





1. **Circular Queue**

**CODE**

#include<stdio.h>

#include<stdlib.h>

void enqueue();

void display();

void dequeue();

void isEmpty();

void isFull();

int a[10],i=0,e,n,t,front=-1,rear=-1;

main()

{

printf(" 17BCE0042\n");

char option='T';

while(option=='T')

{

printf("\n");

printf("Enter command (17BCE0042)\n");

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Display\n");

printf("4. isEmpty\n");

printf("5. isFull\n");

printf("6. Peek\n");

printf("7. Exit\n");

scanf("%d",&n);

switch(n)

{

case 1:

enqueue();

display();

break;

case 2:

dequeue();

display();

break;

case 3:

display();

break;

case 4:

isEmpty();

break;

case 5:

isFull();

display();

break;

case 6:

printf("\nPeek is %d\n",a[rear]);

break;

case 7:

printf(" Thankyou!!\n");

exit(0);

case 8:

printf("\n%d",front);

}

}

}

void dequeue()

{

if (rear==-1 || front==-1 /\*|| front>rear\*/)

{

printf("Sorry\nCircular queue is Empty\n");

return;

}

printf("Done\n");

if(front==4)

{

front=-1;

}

else if(front==rear)

{

front==-1;

rear==-1;

}

else{

front++;

}

}

void enqueue()

{

if (rear+1==front)

{

printf("Sorry\nCQueue is already full\n");

return;

}

if (front==0 && rear==4)

{

printf("Sorry\nCQueue is already full\n");

return;

}

if (rear==4)

{

rear=-1;

}

if(front==-1)

{

front++;

}

/\*if(rear+1==front)

{

front=front+1;

}\*/

printf("Enter number\n");

scanf("%d",&e);

a[rear+1]=e;

rear++;

}

void display()

{

printf("\nStatus\n");

if(front==-1 && rear==-1)

{

return;

}

if(rear<front)

{

for (i=front;i<=4;i++)

{

printf("%d\n",a[i]);

}

for (i=0;i<=rear;i++)

{

printf("%d\n",a[i]);

}

}

else{

for (i=front;i<=rear;i++)

{

printf("%d\n",a[i]);

}

}

}

void isEmpty()

{

if (rear==-1)

{

printf("Yes, it is empty");

}

else

{

printf("No, it is not empty");

}

}

void isFull()

{

if (rear==4)

{

printf("Stack is FULL!!");

}

else

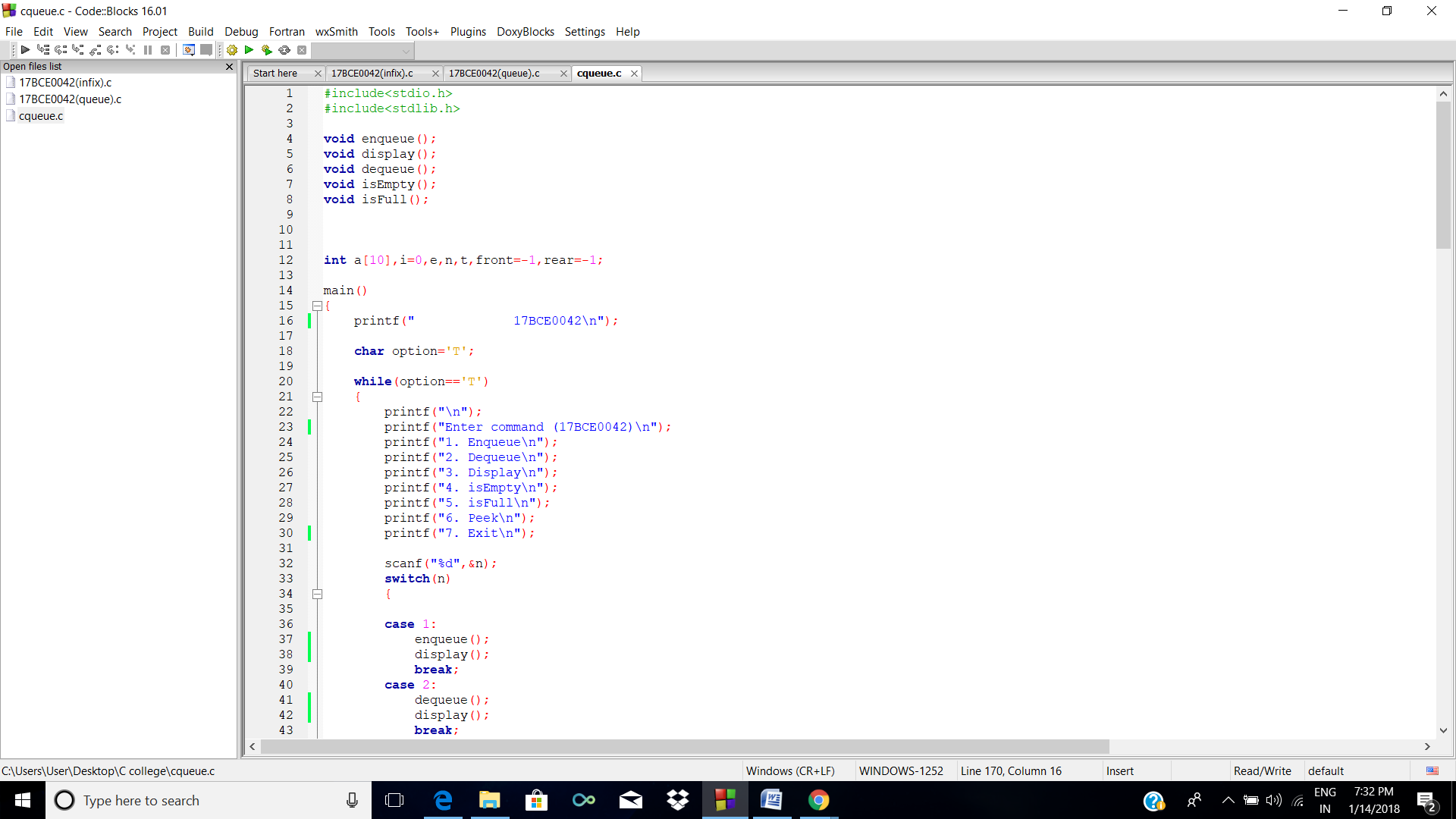
{

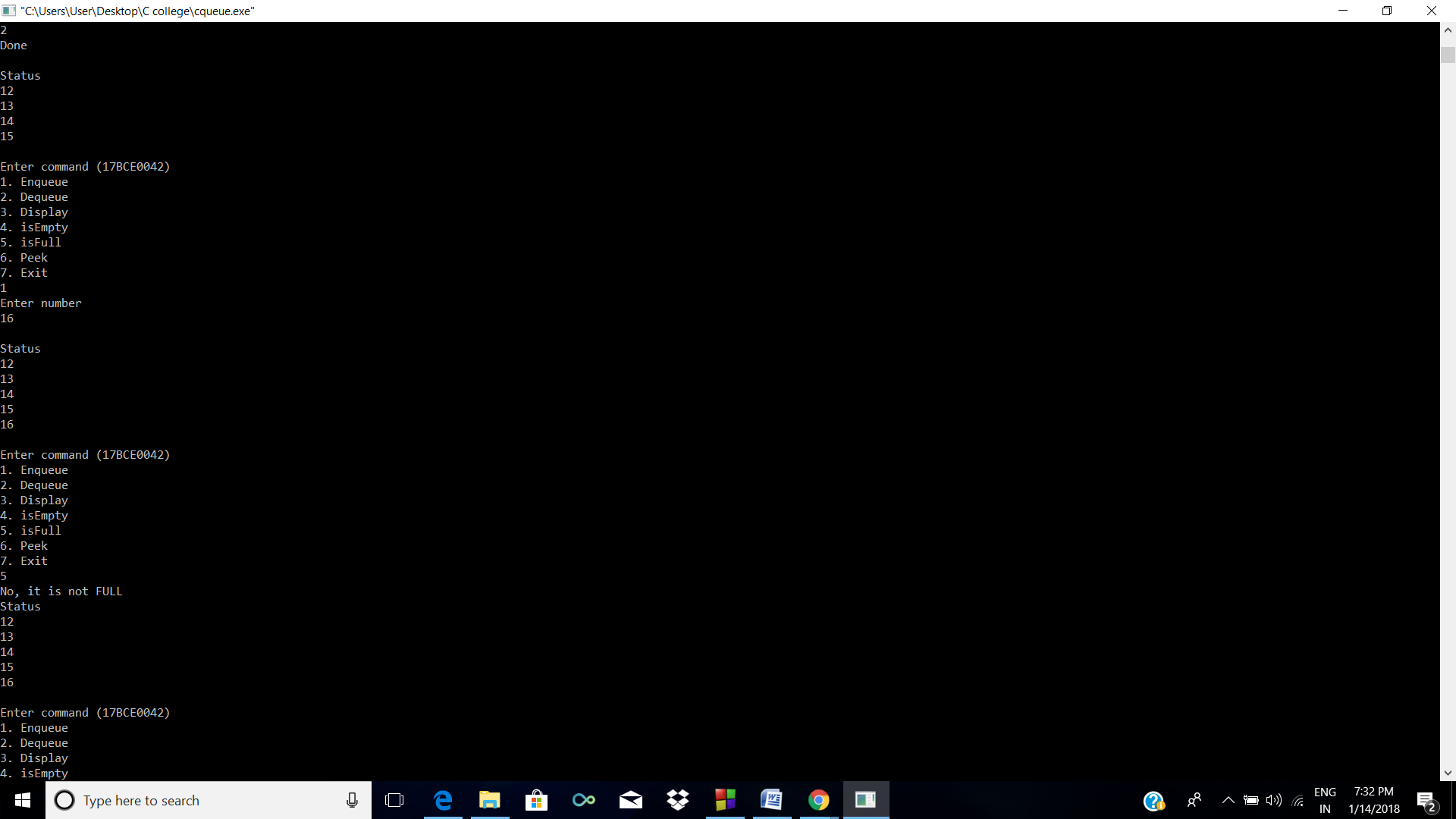
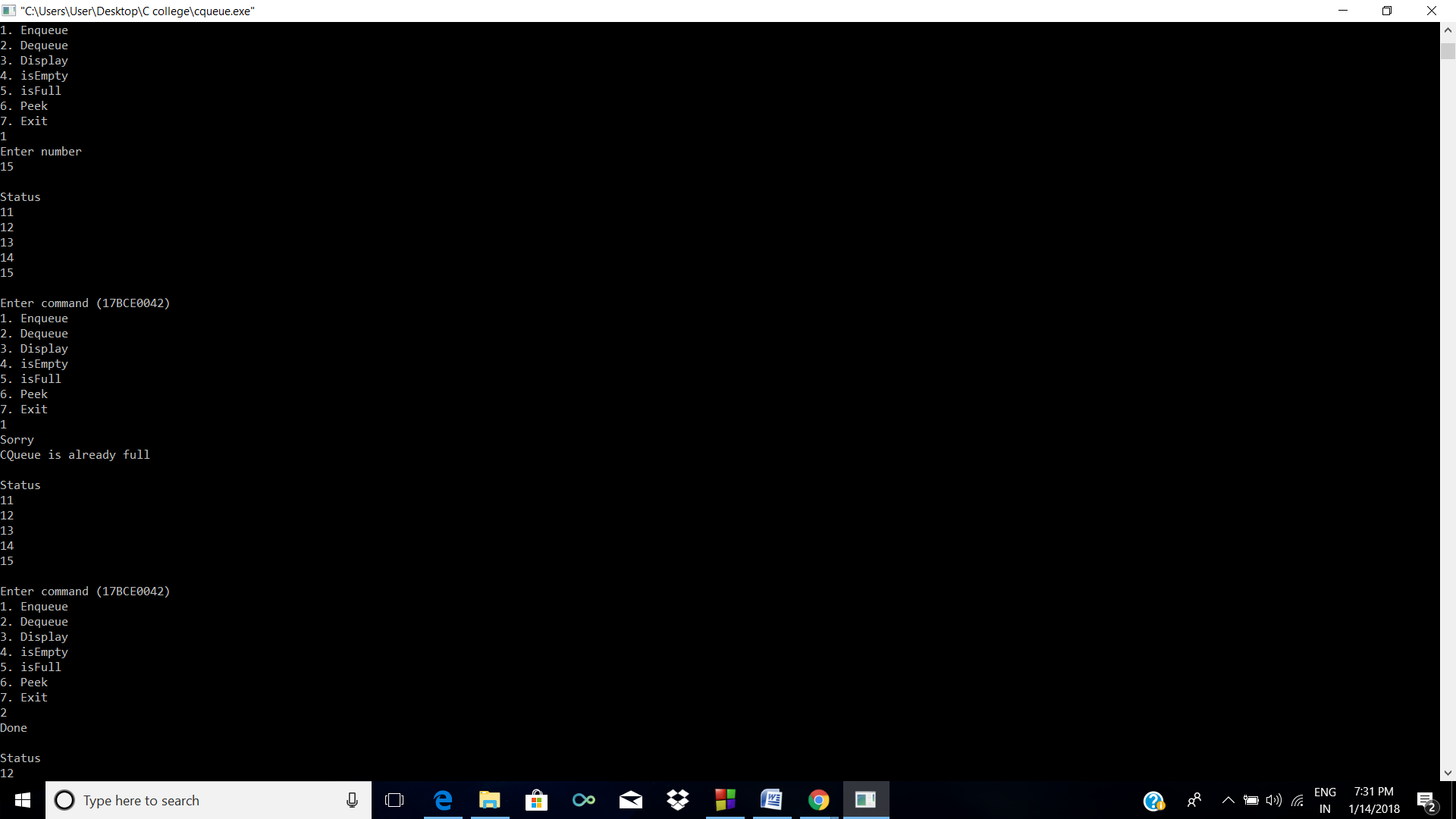
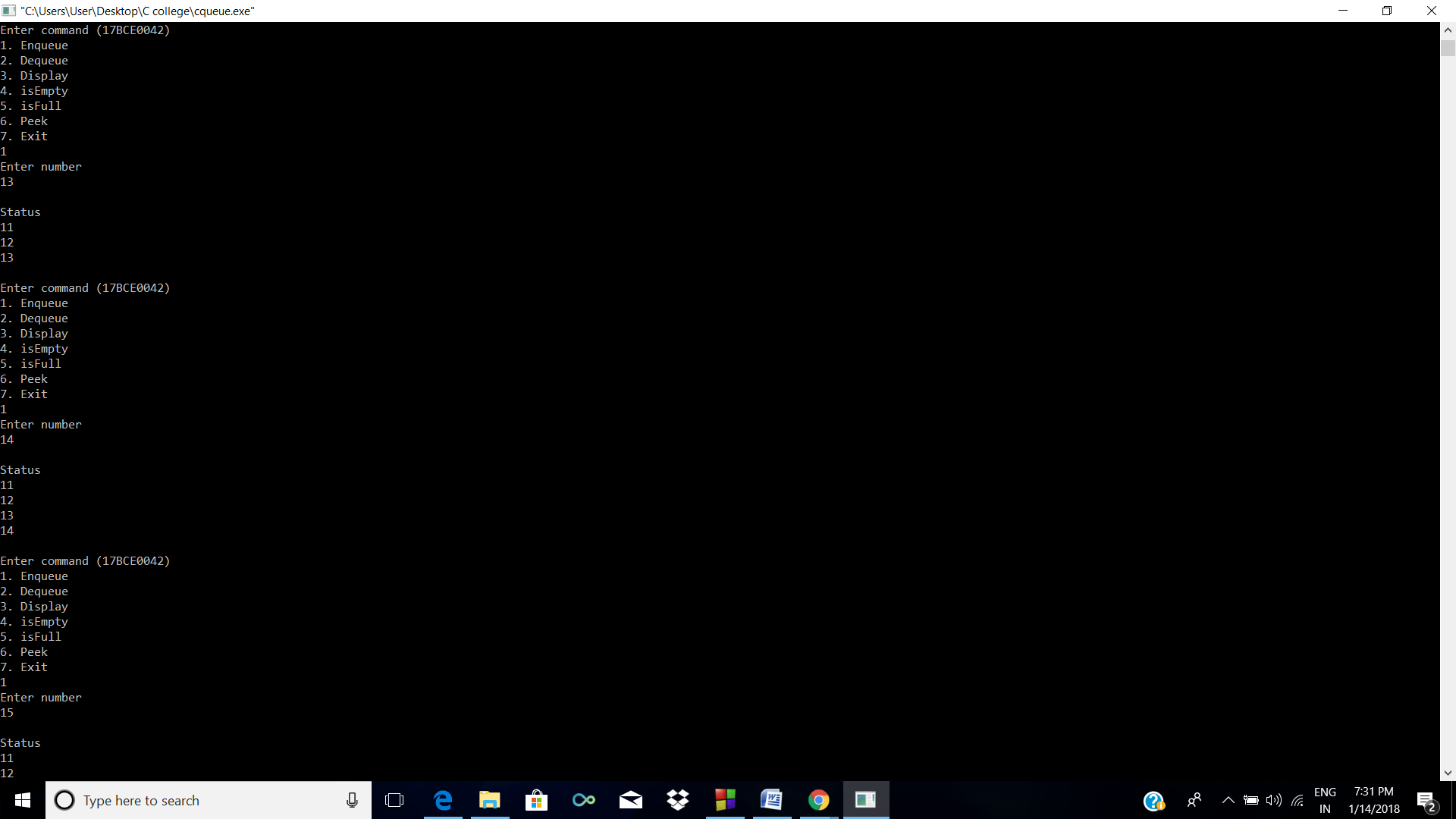
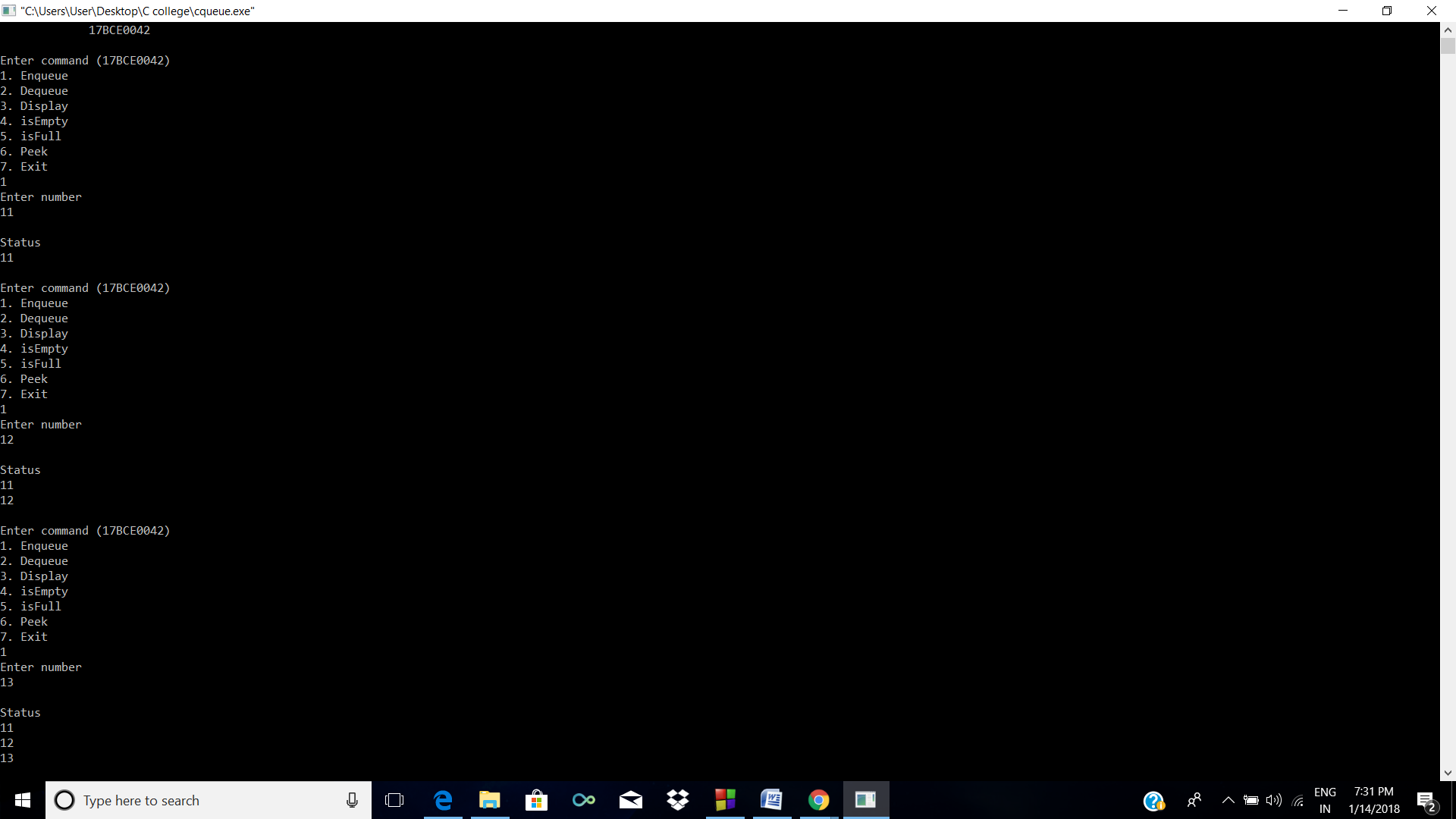
printf("No, it is not FULL");

}

}

**SCREENSHOT**





**6.Double Ended Queue**

**CODE**

#include<stdio.h>

#include<stdlib.h>

void enqueue\_f();

void display();

void dequeue\_f();

void dequeue\_r();

void enqueue\_r();

void isEmpty();

void isFull();

int a[10],i=0,e,n,t,front=-1,rear=-1;

main()

{

printf(" 17BCE0042\n");

char option='T';

while(option=='T')

{

printf("\n");

printf("Enter command (17BCE0042)\n");

printf("1. InsertInFront\n");

printf("2. InsertInBack\n");

printf("3. DeleteInFront\n");

printf("4. DeleteInBack\n");

printf("5. Display\n");

printf("6. isEmpty\n");

printf("7. isFull\n");

printf("8. getFront\n");

printf("9. getRear\n");

printf("10. Exit\n\n");

scanf("%d",&n);

switch(n)

{

case 1:

enqueue\_f();

display();

break;

case 2:

enqueue\_r();

display();

break;

case 3:

dequeue\_f();

display();

break;

case 4:

dequeue\_r();

display();

break;

case 5:

display();

break;

case 6:

isEmpty();

break;

case 7:

isFull();

display();

break;

case 8:

printf("\nFront element is %d\n",a[front]);

break;

case 9:

printf("\nRear element is %d\n",a[rear]);

break;

case 10:

printf(" Thankyou!!\n");

exit(0);

}

}

}

void dequeue\_f()

{

if (front==-1)

{

printf("Sorry\nQueue is Empty\n");

return;

}

printf("Done\n");

front++;

}

void dequeue\_r()

{

if (rear==-1)

{

printf("Sorry\nQueue is Empty\n");

return;

}

printf("Done\n");

rear--;

}

void enqueue\_f()

{

if (front==0)

{

printf("Sorry\nQueue is already full\n");

return;

}

if(front==-1)

{

front++;

rear++;

printf("Enter number\n");

scanf("%d",&e);

a[front]=e;

return;

}

printf("Enter number\n");

scanf("%d",&e);

a[front-1]=e;

front--;

}

void enqueue\_r()

{

if (rear==4)

{

printf("Sorry\nQueue is already full\n");

return;

}

if(front==-1)

{

front++;

rear++;

}

printf("Enter number\n");

scanf("%d",&e);

a[rear+1]=e;

rear++;

}

void display()

{

printf("\nStatus\n");

for (i=front;i<=rear;i++)

{

printf("%d\n",a[i]);

}

}

void isEmpty()

{

if (rear==-1)

{

printf("Yes, it is empty");

}

}

void isFull()

{

if (rear==4)

{

printf("Stack is FULL!!");

}

}

**SCREENSHOT**

