Contents

[Bài thực hành số 1 – Tuần 29 2](#_Toc67755720)

[Bài 1.1**.**Viết một chương trình C nhập vào 3 số nguyên. Thiết lập một con trỏ để lần lượt trỏ tới từng số nguyên và hiển thị kết quả giá trị tham chiếu ngược của con trỏ. 2](#_Toc67755721)

[Bài 1.2. Viết chương trình in ra địa chỉ của 5 phần tử đầu tiên trong mảng được định nghĩa sau đây: int a[7]= {13, -355, 235, 47, 67, 943, 1222}; 3](#_Toc67755722)

[Bài 1.3. Viết chương trình yêu cầu nhập giá trị cho 3 biến số nguyên x, y, z kiểu int. Sau đó sử dụng duy nhất một con trỏ để cộng giá trị của mỗi biến thêm 100. 5](#_Toc67755723)

[Bài 1.4. Viết hàm countEven(int\*, int) nhận một mảng số nguyên và kích thước của mảng, trả về số lượng số chẵn trong mảng??? 6](#_Toc67755724)

[Bài 1.5. Viết hàm trả về con trỏ trỏ tới giá trị lớn nhất của một mảng các số double. Nếu mảng rỗng hãy trả về NULL. 7](#_Toc67755725)

[Bài 1.6**.** Viết hàm đảo ngược một mảng các số nguyên theo hai cách: dùng chỉ số và dùng con trỏ. 8](#_Toc67755726)

[Bài 1.7**.**Viết chương trình nhập vào một mảng các số nguyên với số lượng các phần tử nhập từ bàn phím. Sau đó sắp xếp mảng theo thứ tự tăng dần. Hiển thị danh sách mảng trước và sau khi sắp xếp. 10](#_Toc67755727)

[Bài 1.8**.**Viết chương trình nhập vào một ma trận 2 chiều kích thước m\*n với m và n nhập từ bàn phím. Sau đó đưa ra tổng các phần tử chẵn của ma trận đó. 12](#_Toc67755728)

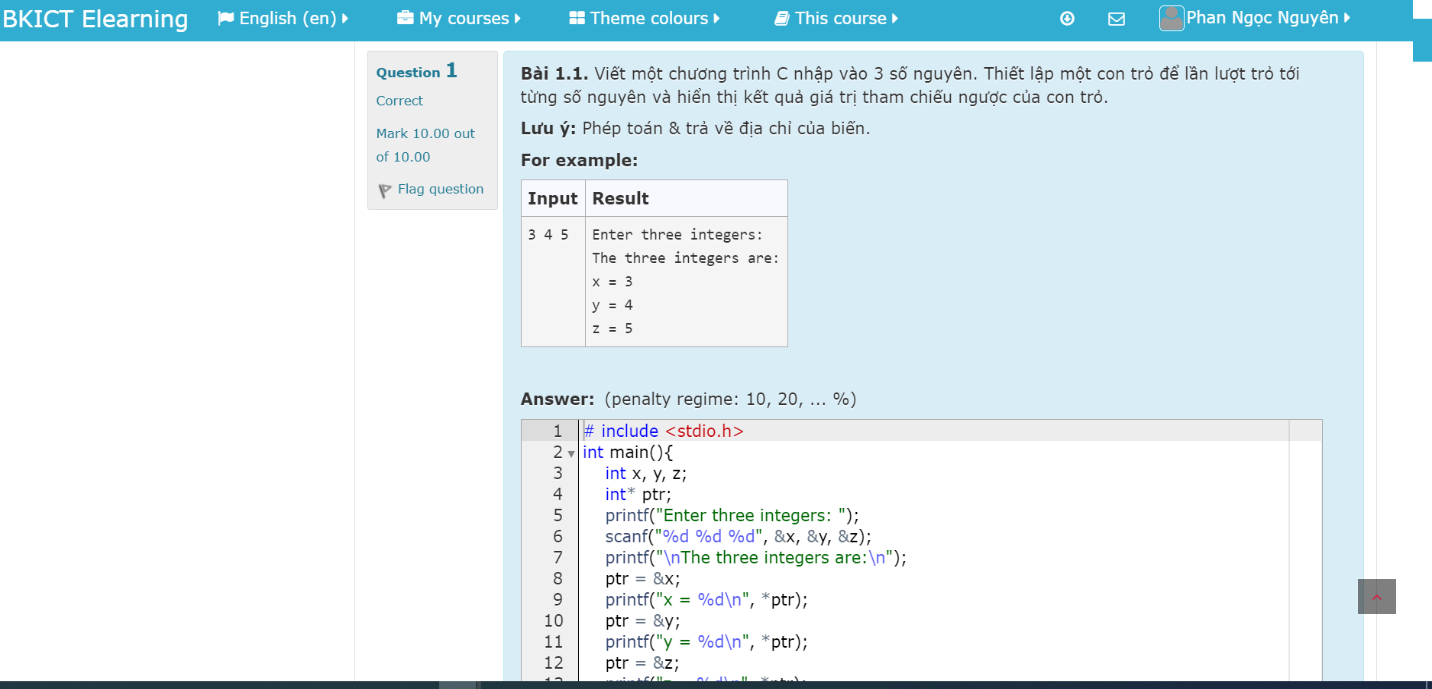
[Bài 1.9: Viết chương trình in ra tất cả các dãy con của một dãy cho trước. 15](#_Toc67755729)

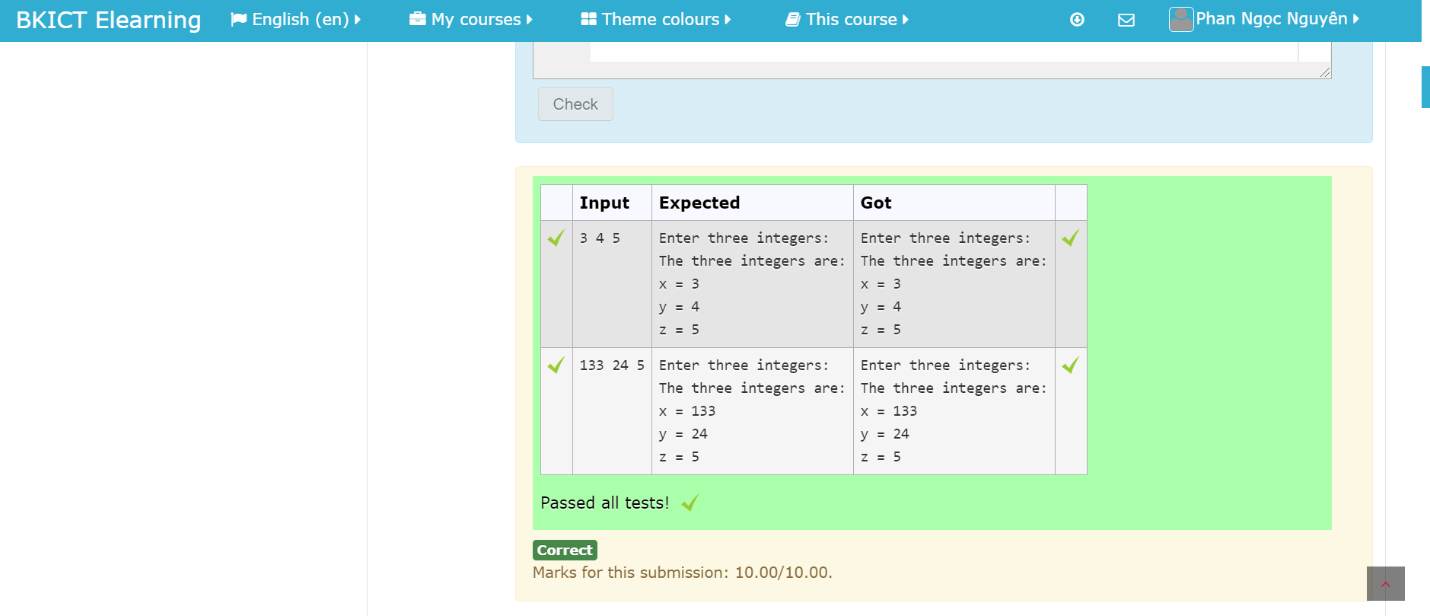
[Bài 1.10: Viết chương trình nhập vào 2 ma trận vuông cùng kích thước n\*n, trong đó n nhập từ bàn phím. Sau đó tính tổng và tích của hai ma trận đó và đưa kết quả ra màn hình. 16](#_Toc67755730)

# Bài thực hành số 1 – Tuần 29

## Bài 1.1**.**Viết một chương trình C nhập vào 3 số nguyên. Thiết lập một con trỏ để lần lượt trỏ tới từng số nguyên và hiển thị kết quả giá trị tham chiếu ngược của con trỏ.

**Lưu ý:** Phép toán & trả về địa chỉ của biến.





Code:

# include <stdio.h>

int main(){

int x, y, z;

int\* ptr;

printf("Enter three integers: ");

scanf("%d %d %d", &x, &y, &z);

printf("\nThe three integers are:\n");

ptr = &x;

printf("x = %d\n", \*ptr);

ptr = &y;

printf("y = %d\n", \*ptr);

ptr = &z;

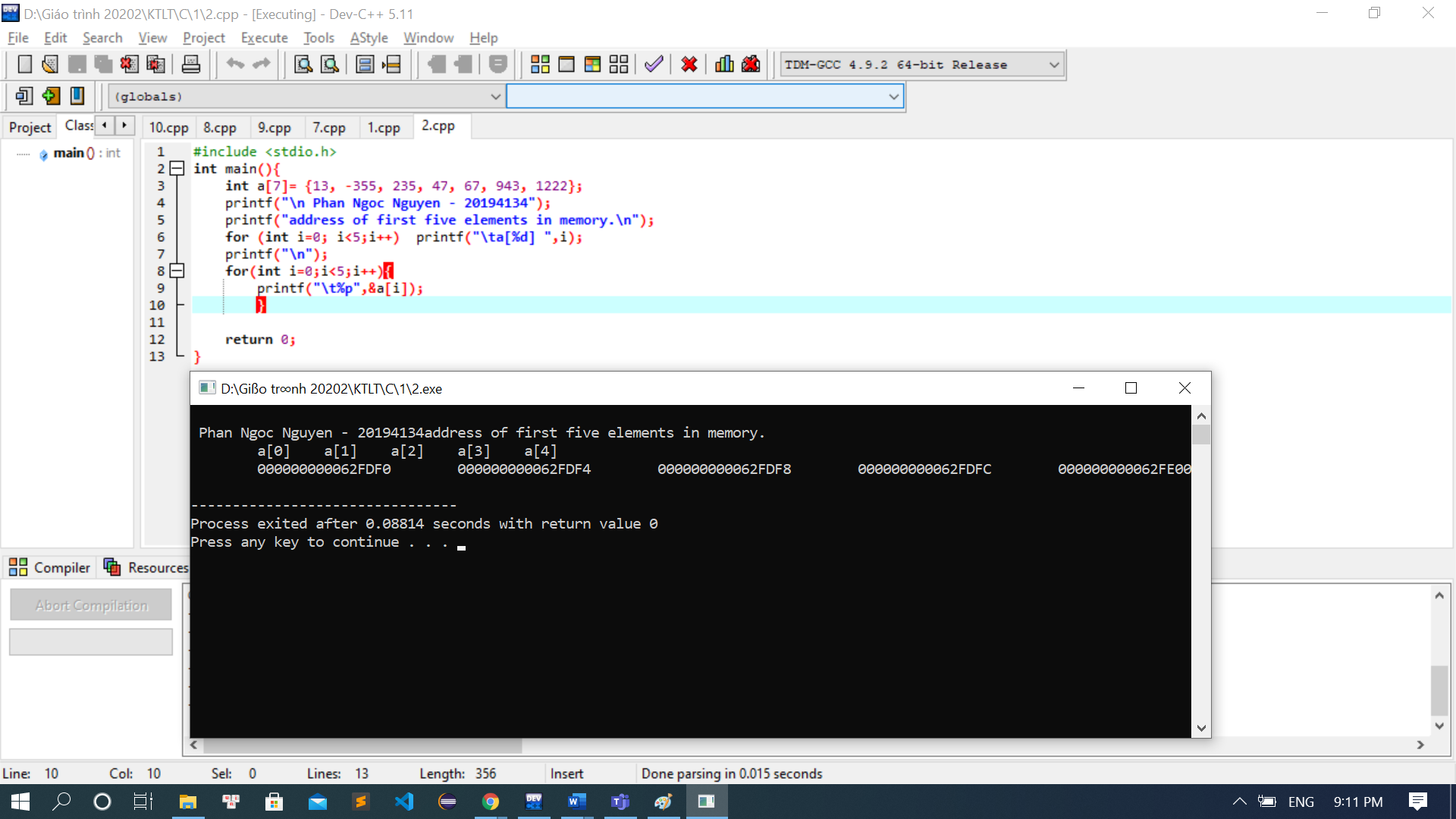
printf("z = %d\n", \*ptr)

return 0;

}

## Bài 1.2. Viết chương trình in ra địa chỉ của 5 phần tử đầu tiên trong mảng được định nghĩa sau đây: int a[7]= {13, -355, 235, 47, 67, 943, 1222};

**Lưu ý:**  
**Để in địa chỉ con trỏ các bạn sử dụng ký tự định dạng %p**   
**Để lấy địa chỉ của một biến ta có thể dùng phép toán &**



Code:

#include <stdio.h>

int main(){

int a[7]= {13, -355, 235, 47, 67, 943, 1222};

printf("\n Phan Ngoc Nguyen - 20194134");

printf("address of first five elements in memory.\n");

for (int i=0; i<5;i++) printf("\ta[%d] ",i);

printf("\n");

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

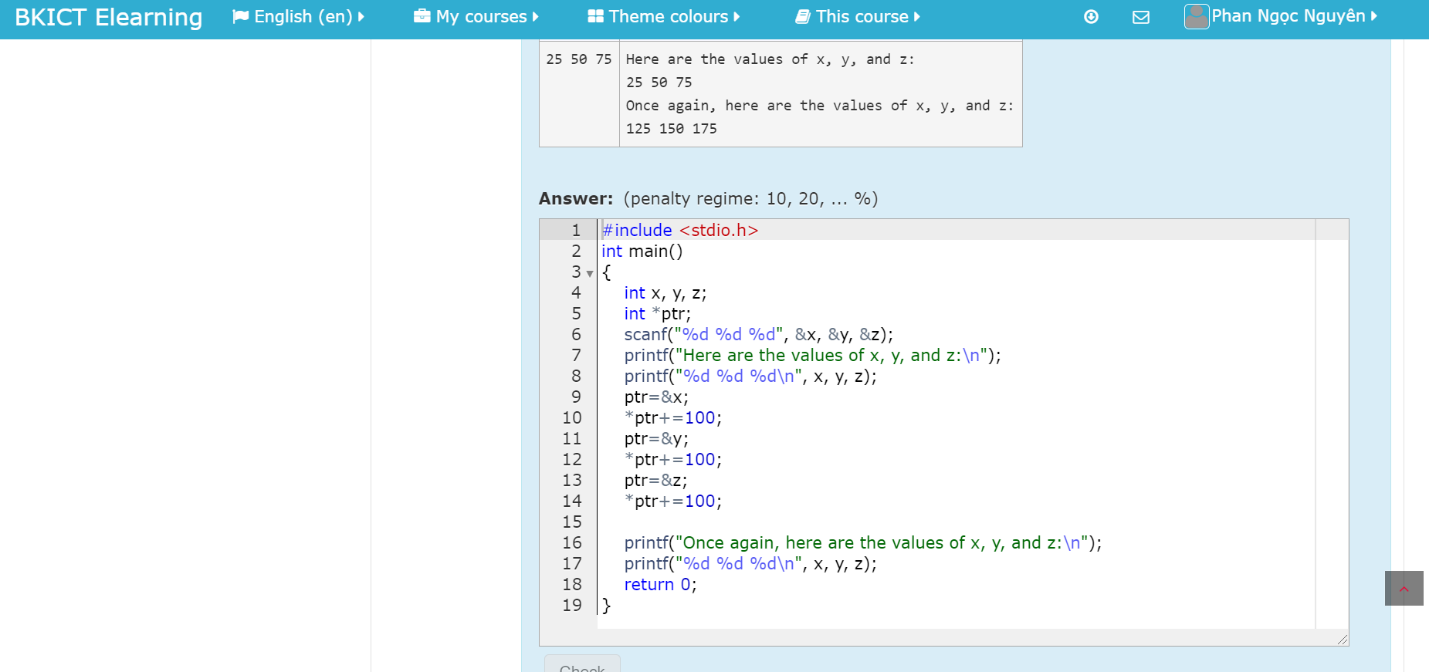
printf("\t%p",&a[i]);

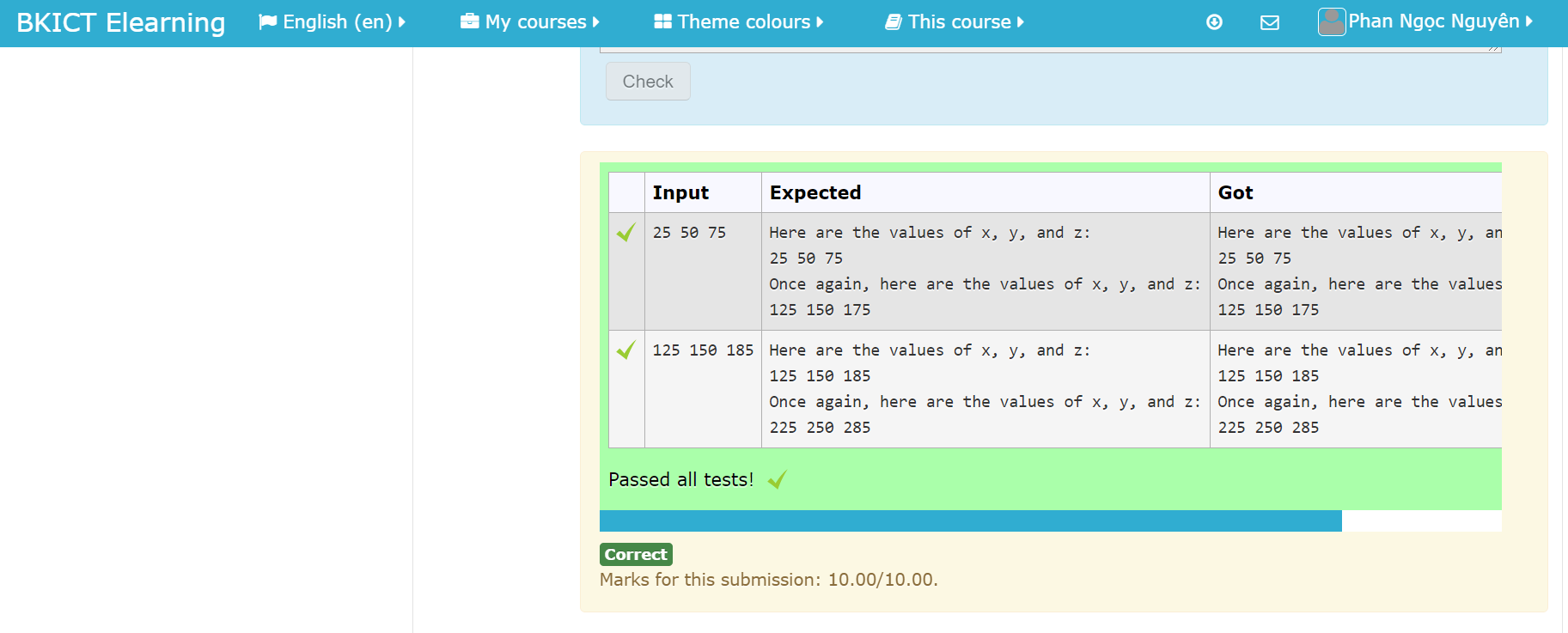
}

return 0;

}

## Bài 1.3. Viết chương trình yêu cầu nhập giá trị cho 3 biến số nguyên x, y, z kiểu int. Sau đó sử dụng duy nhất một con trỏ để cộng giá trị của mỗi biến thêm 100.



Code:

#include <stdio.h>

int main()

{

int x, y, z;

int \*ptr;

scanf("%d %d %d", &x, &y, &z);

printf("Here are the values of x, y, and z:\n");

printf("%d %d %d\n", x, y, z);

ptr=&x;

\*ptr+=100;

ptr=&y;

\*ptr+=100;

ptr=&z;

\*ptr+=100;

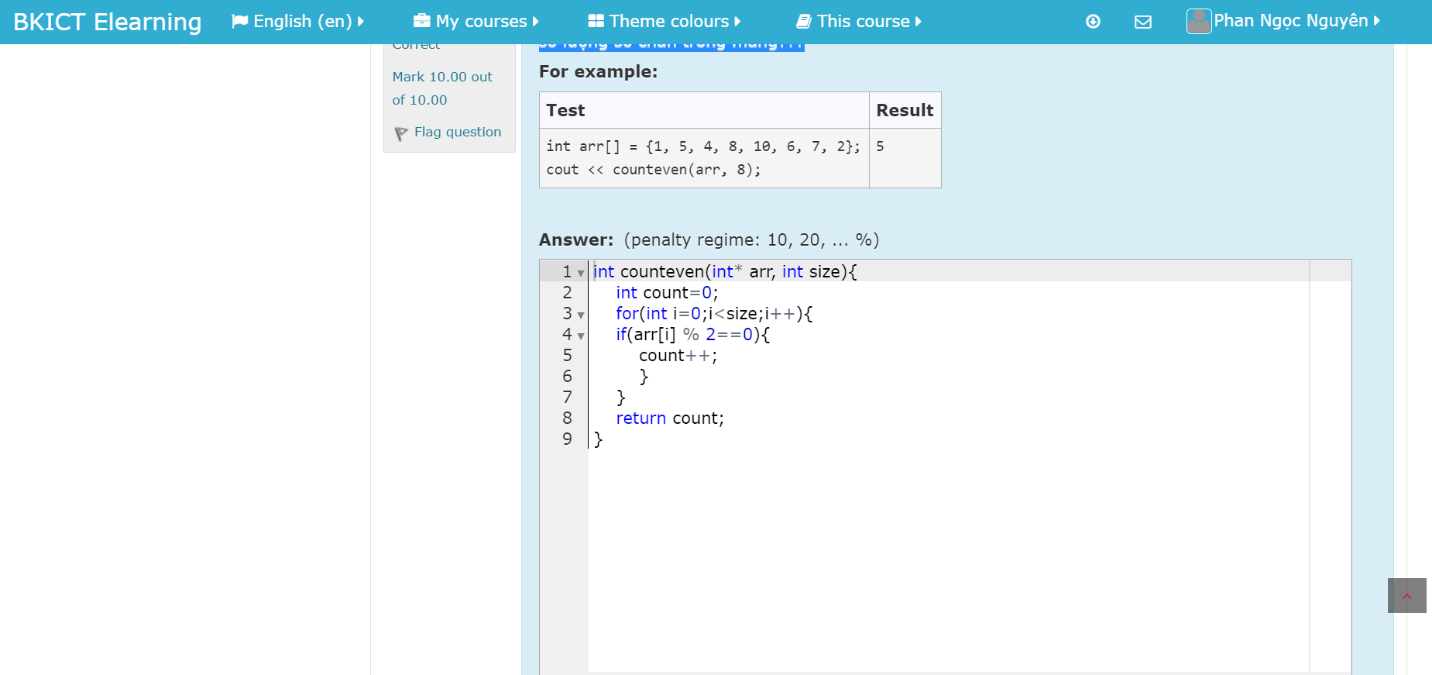
printf("Once again, here are the values of x, y, and z:\n");

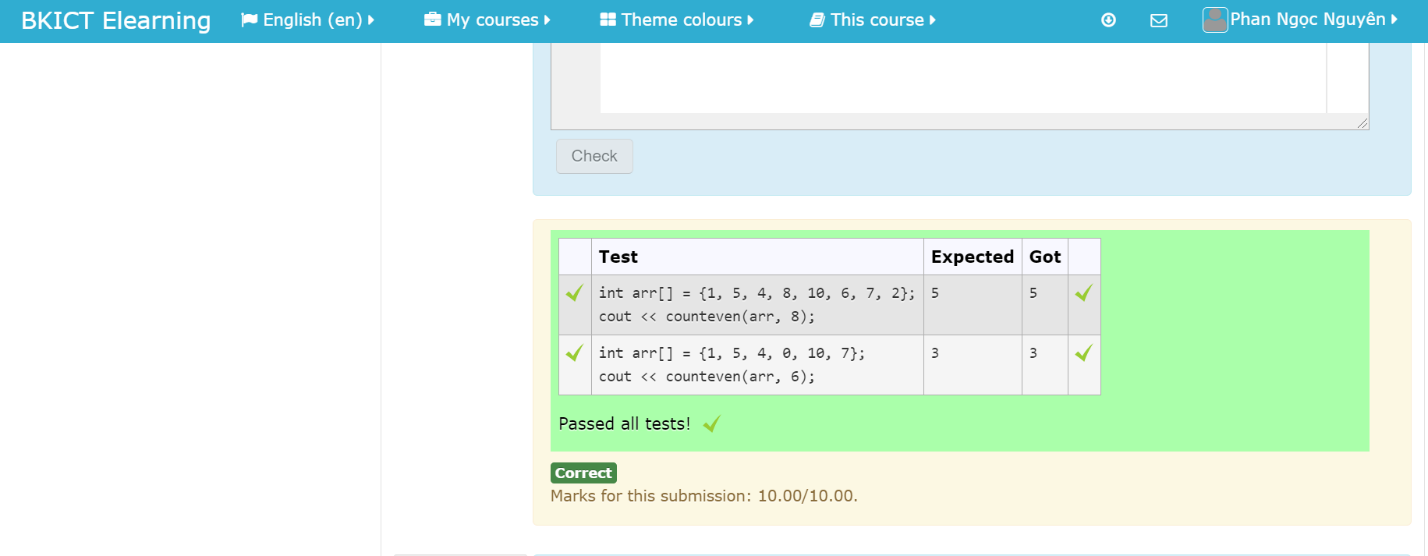
printf("%d %d %d\n", x, y, z);

return 0;

}

## Bài 1.4. Viết hàm countEven(int\*, int) nhận một mảng số nguyên và kích thước của mảng, trả về số lượng số chẵn trong mảng???





Code:

#include <stdio.h>

int counteven(int\* arr, int size){

int count=0;

for(int i=0;i<size;i++){

if(arr[i] % 2==0){

count++;

}

}

return count;

}

int main(){

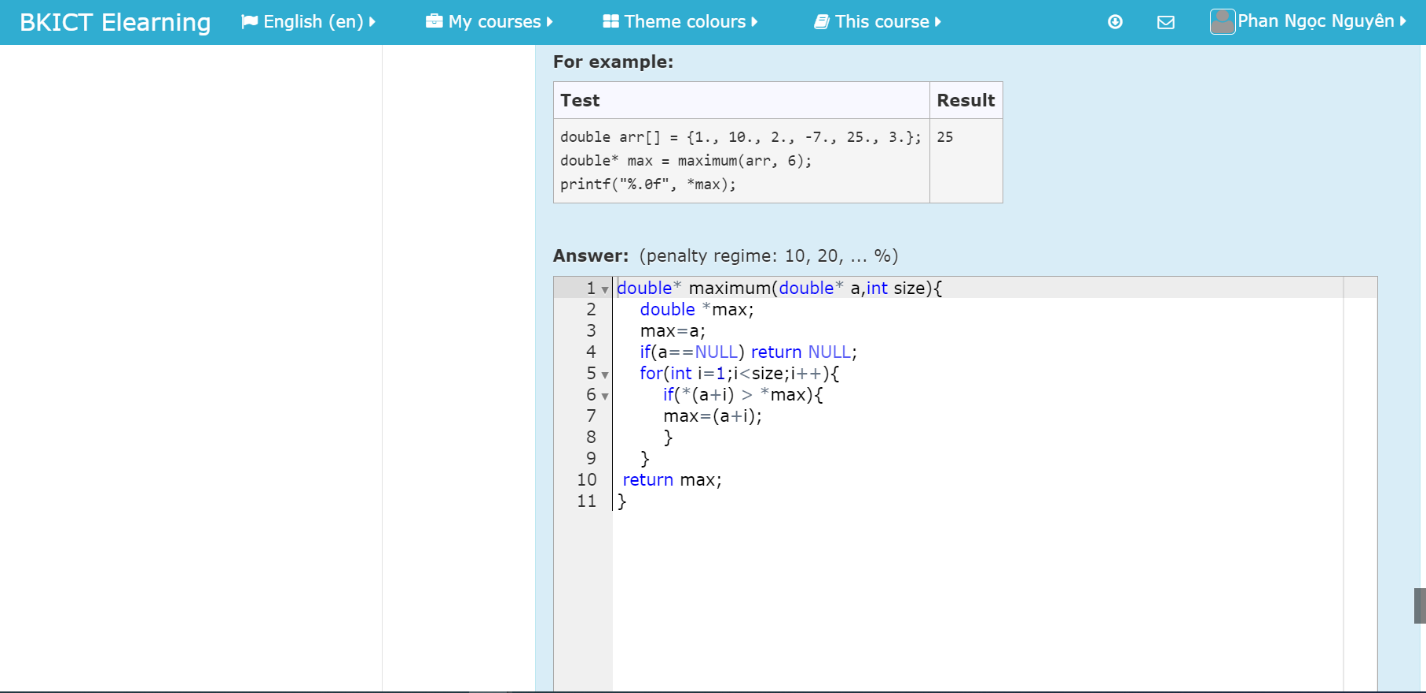
int arr[]={1,5,4,8,10,6,7,2};

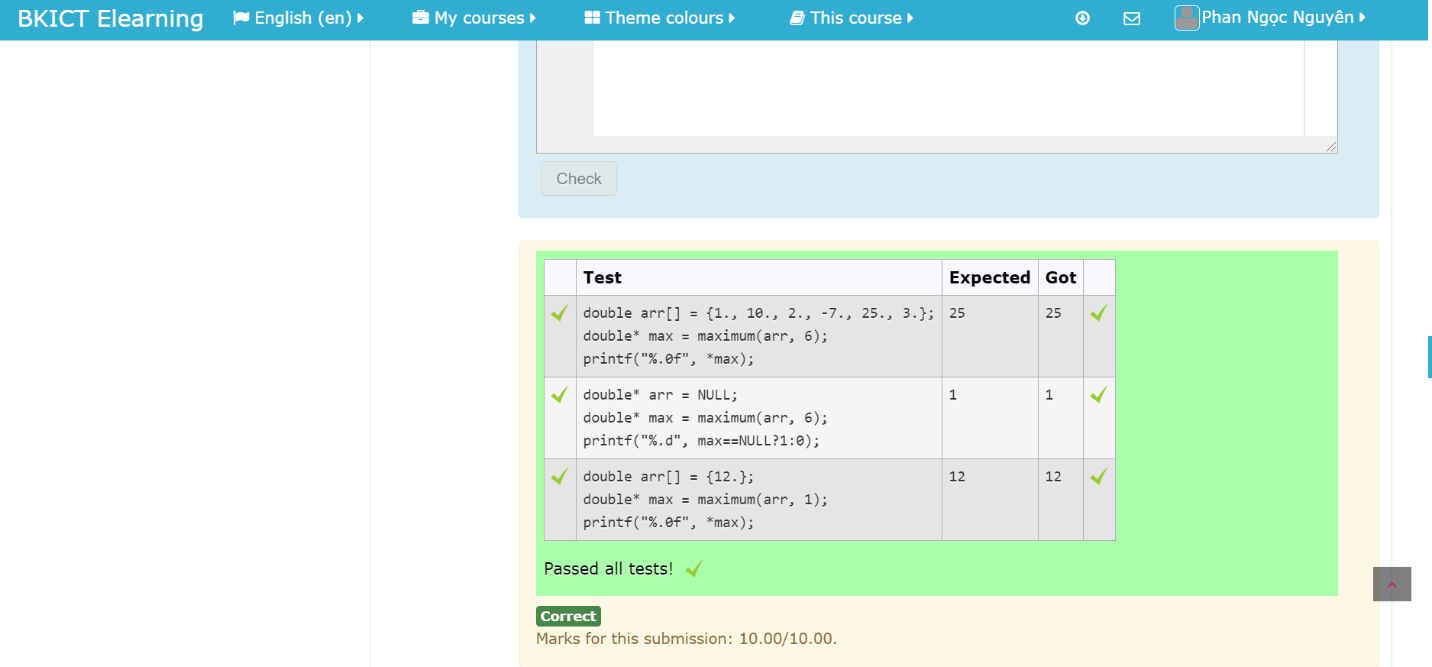
printf("%d ",counteven(arr,8));

return 0;

}

## Bài 1.5. Viết hàm trả về con trỏ trỏ tới giá trị lớn nhất của một mảng các số double. Nếu mảng rỗng hãy trả về NULL.





Code: #include <stdio.h>

double\* maximum(double\* a,int size){

double \*max;

max=a;

if(a==NULL) return NULL;

for(int i=1;i<size;i++){

if(\*(a+i) > \*max){

max=(a+i);

}

}

return max;

}

int main(){

double arr[]={1.,10.,2.,-7.,25.,3.};

double\* max=maximum(arr,6);

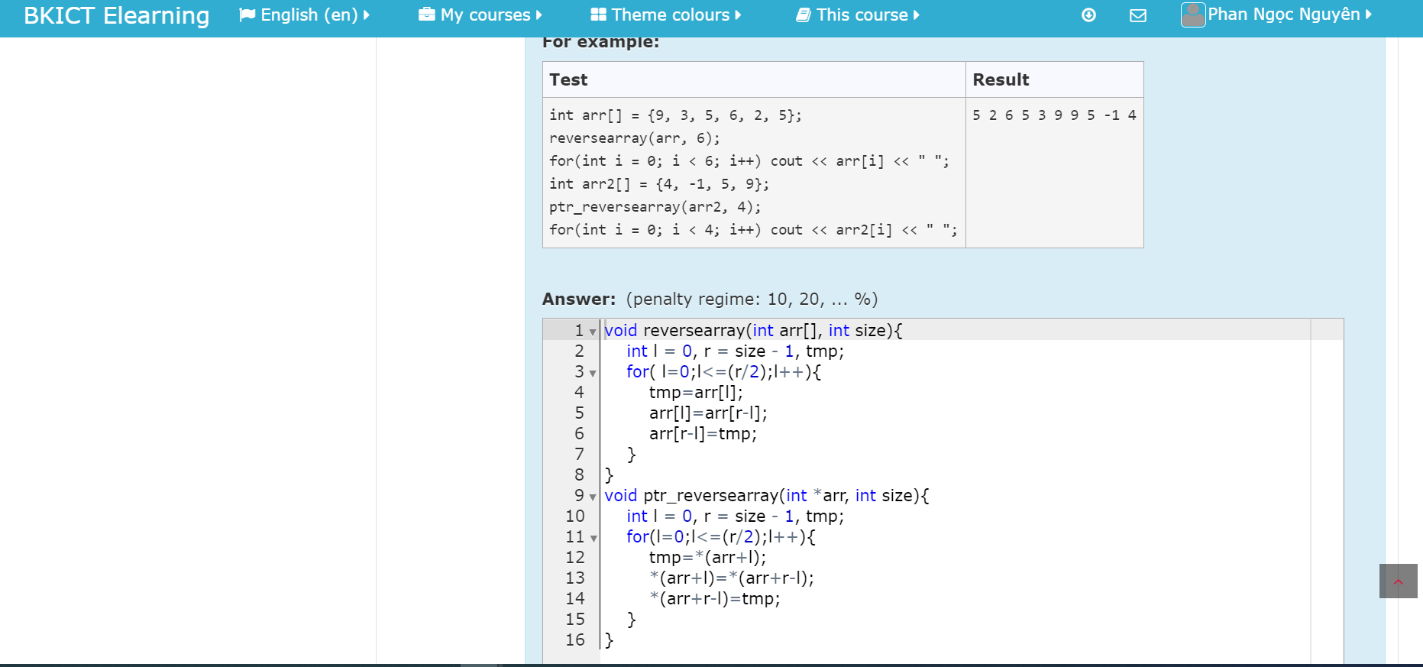
printf("%.0f",\*max);

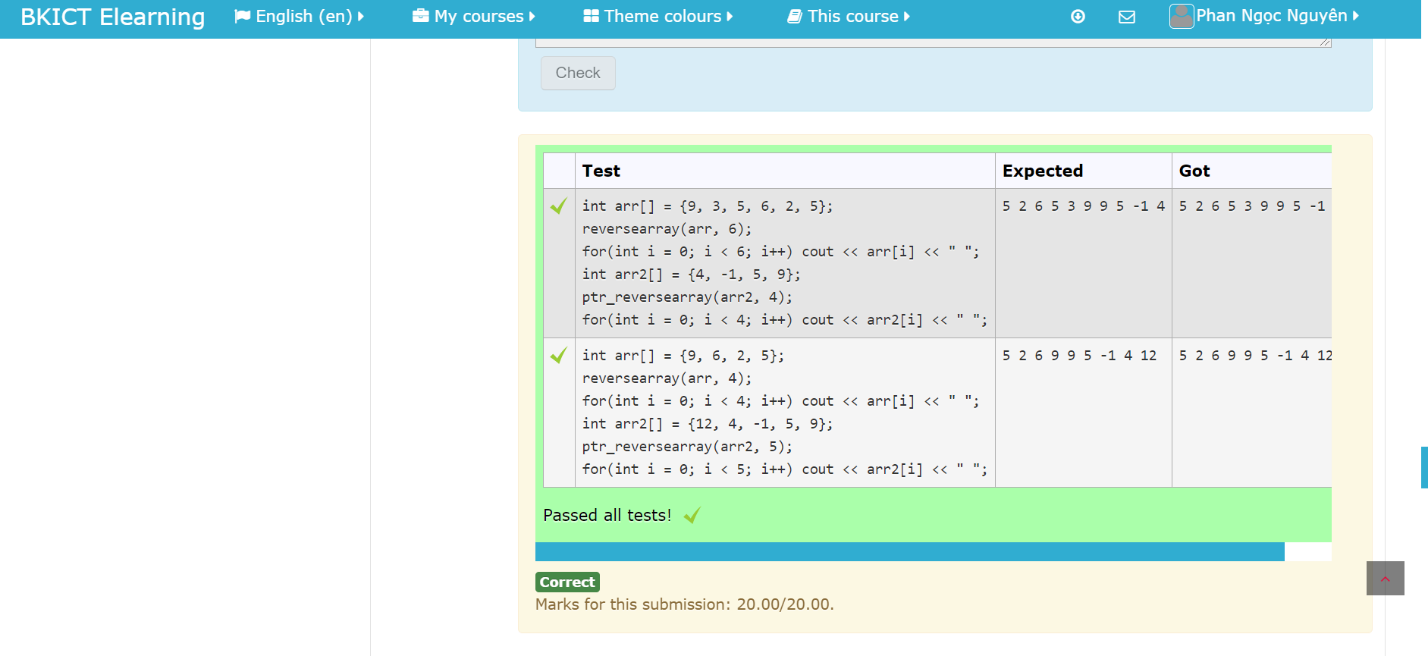
return 0;

}

## Bài 1.6**.** Viết hàm đảo ngược một mảng các số nguyên theo hai cách: dùng chỉ số và dùng con trỏ.

Ví dụ mảng đầu vào là [9, -1, 4, 5, 7] thì kết quả là [7, 5, 4, -1, 9].





Code:

#include <stdio.h>

void reversearray(int arr[], int size){

int l = 0, r = size - 1, tmp;

for( l=0;l<=(r/2);l++){

tmp=arr[l];

arr[l]=arr[r-l];

arr[r-l]=tmp;

}

}

void ptr\_reversearray(int \*arr, int size){

int l = 0, r = size - 1, tmp;

for(l=0;l<=(r/2);l++){

tmp=\*(arr+l);

\*(arr+l)=\*(arr+r-l);

\*(arr+r-l)=tmp;

}

}

int main(){

int arr[] = {9, 3, 5, 6, 2, 5};

reversearray(arr, 6);

for(int i = 0; i < 6; i++){

printf("%d ",arr[i]);

};

printf("\n");

int arr2[] = {4, -1, 5, 9};

ptr\_reversearray(arr2, 4);

for(int i = 0; i < 4; i++){

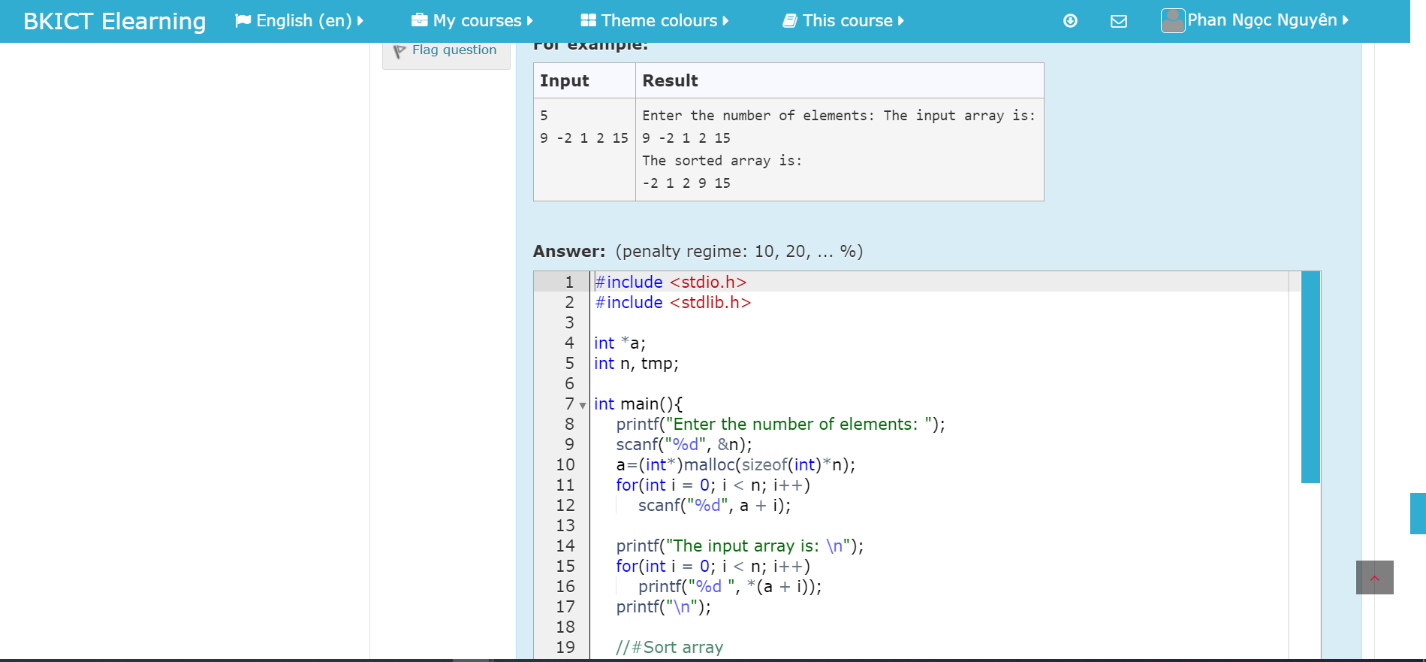
printf("%d ",arr2[i]);

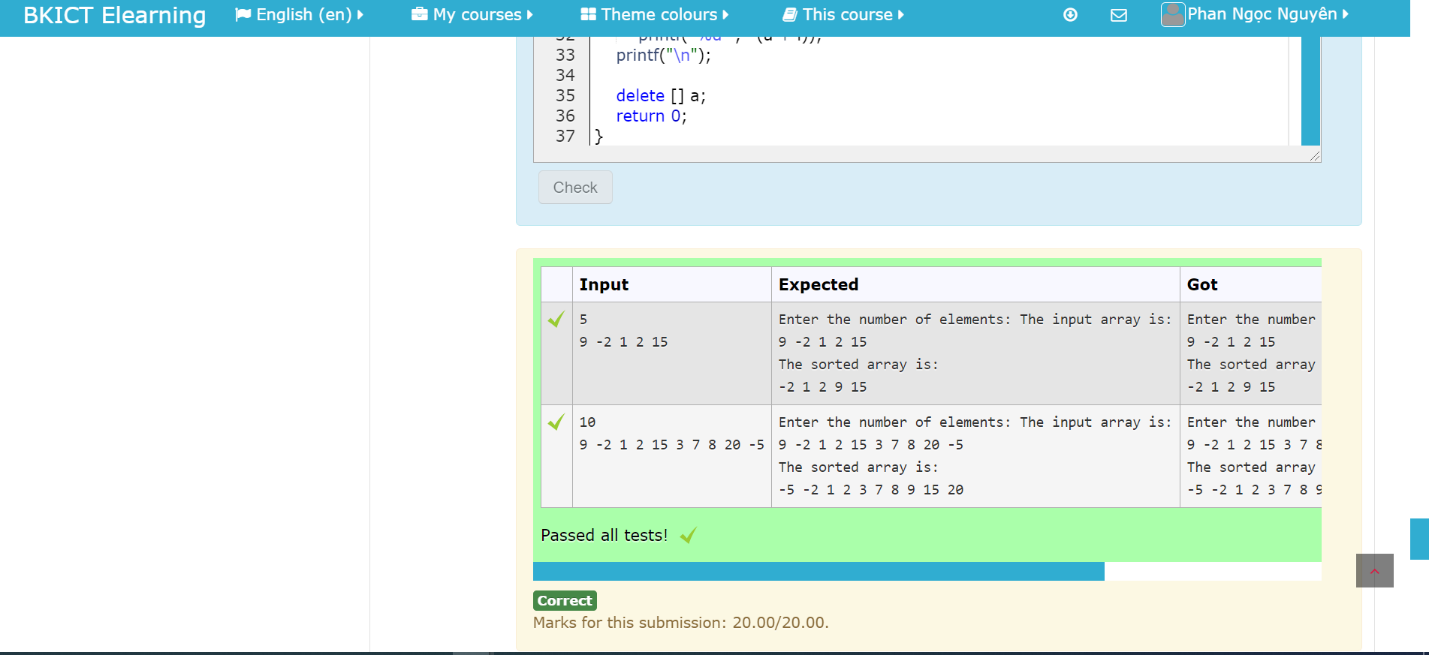
};

}

Bài 1.7**.**Viết chương trình nhập vào một mảng các số nguyên với số lượng các phần tử nhập từ bàn phím. Sau đó sắp xếp mảng theo thứ tự tăng dần. Hiển thị danh sách mảng trước và sau khi sắp xếp.

Yêu cầu chỉ sử dụng con trỏ để truy cập mảng, không truy cập theo index mảng.





Code:

#include <stdio.h>

#include <stdlib.h>

int \*a;

int n, tmp;

int main(){

printf("Enter the number of elements: ");

scanf("%d", &n);

a=(int\*)malloc(sizeof(int)\*n);

for(int i = 0; i < n; i++)

scanf("%d", a + i);

printf("The input array is: \n");

for(int i = 0; i < n; i++)

printf("%d ", \*(a + i));

printf("\n");

//#Sort array

for(int i=0;i<n-1;i++){

for(int j=i;j<n;j++){

if(\*(a+i) > \*(a+j)){

tmp=\*(a+i);

\*(a+i)=\*(a+j);

\*(a+j)=tmp;

}

}

}

printf("The sorted array is: \n");

for(int i = 0; i < n; i++)

printf("%d ", \*(a + i));

printf("\n");

delete [] a;

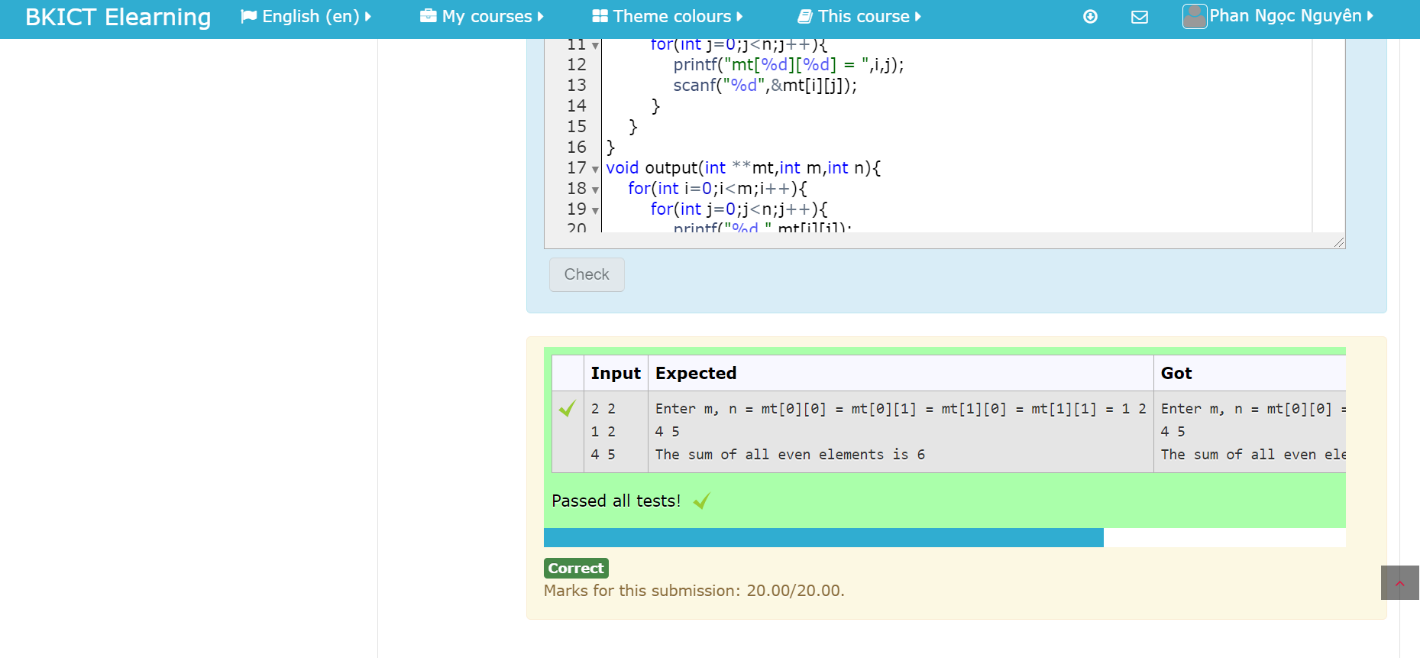
return 0;

}

## Bài 1.8**.**Viết chương trình nhập vào một ma trận 2 chiều kích thước m\*n với m và n nhập từ bàn phím. Sau đó đưa ra tổng các phần tử chẵn của ma trận đó.

**Lưu ý:** Khi viết hàm cấp phát bộ nhớ cho một ma trận hai chiều biểu diễn bởi con trỏ int \*\*mt, nếu ta truyền con trỏ theo kiểu địa chỉ void allocate\_mem(int \*\*mt, int m, int n) sẽ dẫn tới việc cấp phát bộ nhớ cho một bản sao của con trỏ \*\*mt. Do đó, sau khi gọi hàm thì con trỏ \*\*mt gốc vẫn không được cấp phát bộ nhớ. Để cấp phát thành công cần truyền con trỏ theo dạng địa chỉ, ví dụ sử dụng con trỏ cấp 3 dạng int \*\*\*mt.





Code:

#include <stdio.h>

#include <stdlib.h>

void allocate\_mem(int\*\*\* mt,int m,int n){

\*mt=new int\*[m];

for(int i=0;i<m;i++){

(\*mt)[i]=new int[n];

}

}

void input(int \*\*mt,int m,int n){

for(int i=0;i<m;i++){

for(int j=0;j<n;j++){

printf("mt[%d][%d] = ",i,j);

scanf("%d",&mt[i][j]);

}

}

}

void output(int \*\*mt,int m,int n){

for(int i=0;i<m;i++){

for(int j=0;j<n;j++){

printf("%d ",mt[i][j]);

}

printf("\n");

}

}

int process(int \*\*mt,int m,int n){

int tong=0;

for(int i=0;i<m;i++){

for(int j=0;j<n;j++){

if(mt[i][j] % 2 ==0){

tong+=mt[i][j];

}

}

}

return tong;

}

void free\_mem(int \*\*mt,int m,int n){

for(int i=0;i<m;i++){

free(mt[i]);

}

free(mt);

}

int main(){

int m,n, \*\*mt;

printf("Enter m, n = ");

scanf("%d%d",&m,&n);

allocate\_mem(&mt,m,n);

input(mt,m,n);

output(mt, m, n);

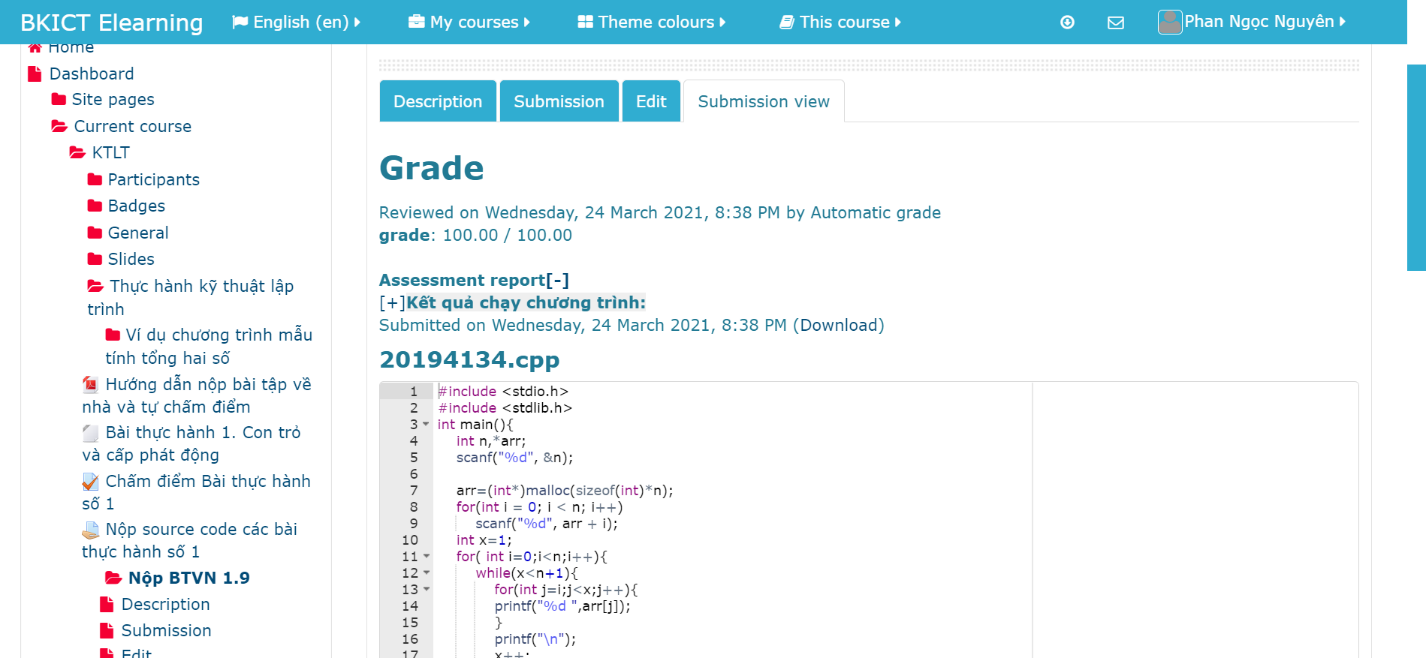
printf("The sum of all even elements is %d", process(mt, m, n));

free\_mem(mt, m, n);

}

Bài tập về nhà:

## Bài 1.9: Viết chương trình in ra tất cả các dãy con của một dãy cho trước.



Code:

#include <stdio.h>

#include <stdlib.h>

int main(){

int n,\*arr;

scanf("%d", &n);

arr=(int\*)malloc(sizeof(int)\*n);

for(int i = 0; i < n; i++)

scanf("%d", arr + i);

int x=1;

for( int i=0;i<n;i++){

while(x<n+1){

for(int j=i;j<x;j++){

printf("%d ",arr[j]);

}

printf("\n");

x++;

}

x=i+2;

}

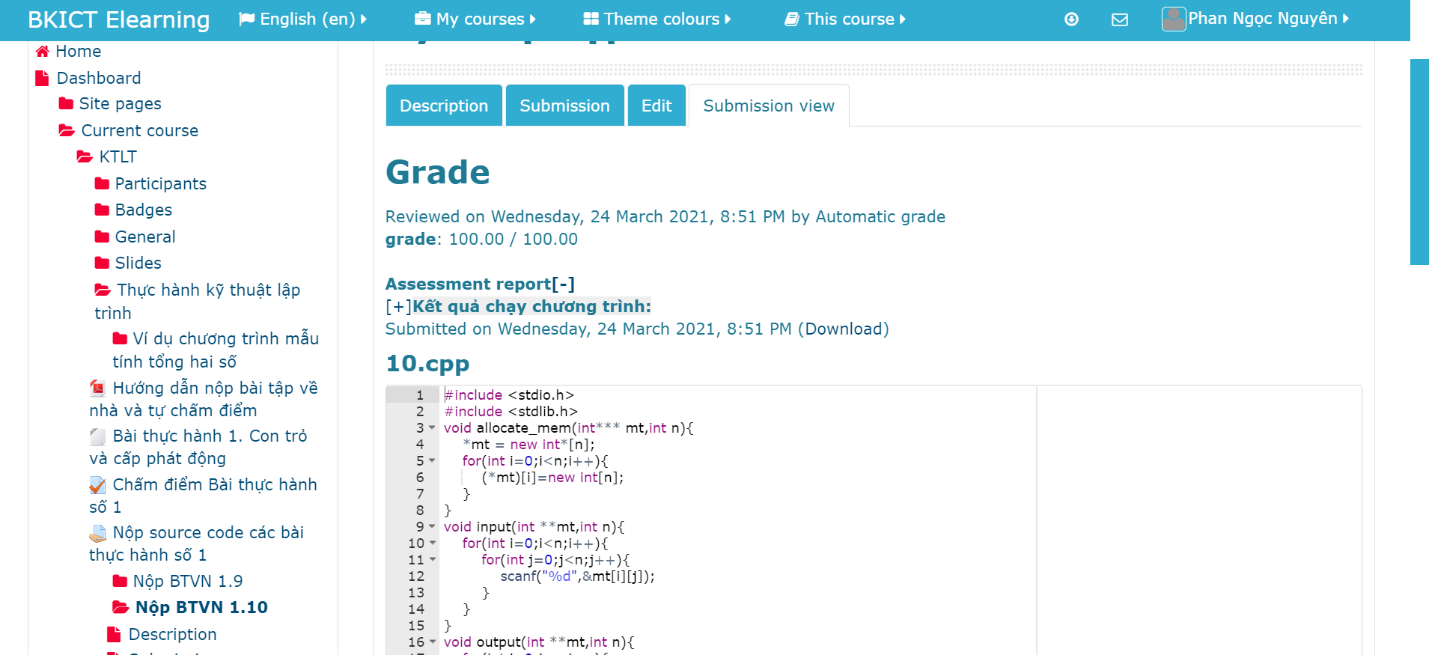
free(arr);

return 0;

}

## Bài 1.10: Viết chương trình nhập vào 2 ma trận vuông cùng kích thước n\*n, trong đó n nhập từ bàn phím. Sau đó tính tổng và tích của hai ma trận đó và đưa kết quả ra màn hình.

Yêu cầu sử dụng cấp phát động để cấp phát bộ nhớ cho các ma trận.



Code:

#include <stdio.h>

#include <stdlib.h>

void allocate\_mem(int\*\*\* mt,int n){

\*mt = new int\*[n];

for(int i=0;i<n;i++){

(\*mt)[i]=new int[n];

}

}

void input(int \*\*mt,int n){

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

scanf("%d",&mt[i][j]);

}

}

}

void output(int \*\*mt,int n){

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

printf("%d ",mt[i][j]);

}

printf("\n");

}

}

int\*\* sumMt(int\*\* mt1,int\*\* mt2,int n){

int\*\* sum;

allocate\_mem(&sum,n);

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

sum[i][j]=mt1[i][j]+mt2[i][j];

}

}

return sum;

}

int\*\* mulMt(int\*\* mt1,int\*\* mt2,int n){

int\*\* mul;

allocate\_mem(&mul,n);

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

mul[i][j]=0;

for(int k=0;k<n;k++){

mul[i][j]+=mt1[i][k]\*mt2[k][j];

}

}

}

return mul;

}

void free\_mem(int \*\*mt,int n){

for(int i=0;i<n;i++){

free(mt[i]);

}

free(mt);

}

int main(){

int n, \*\*mt1,\*\*mt2;

scanf("%d",&n);

allocate\_mem(&mt1,n);

allocate\_mem(&mt2,n);

input(mt1,n);

input(mt2,n);

int \*\*sum=sumMt(mt1,mt2,n);

output(sum,n);

int \*\*mul=mulMt(mt1,mt2,n);

output(mul,n);

free\_mem(mt1,n);

free\_mem(mt2,n);

free\_mem(sum,n);

free\_mem(mul,n);

return 0;

}