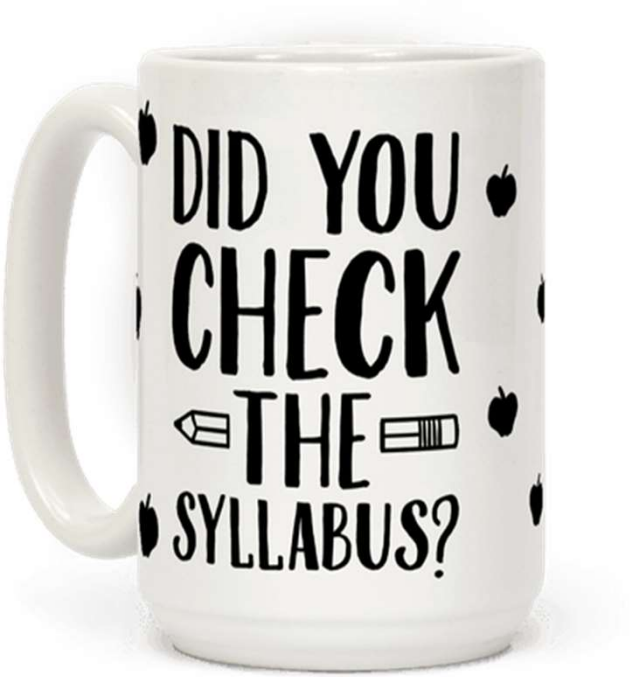
Arrays

19CSE102 Computer Programming

CONTENT

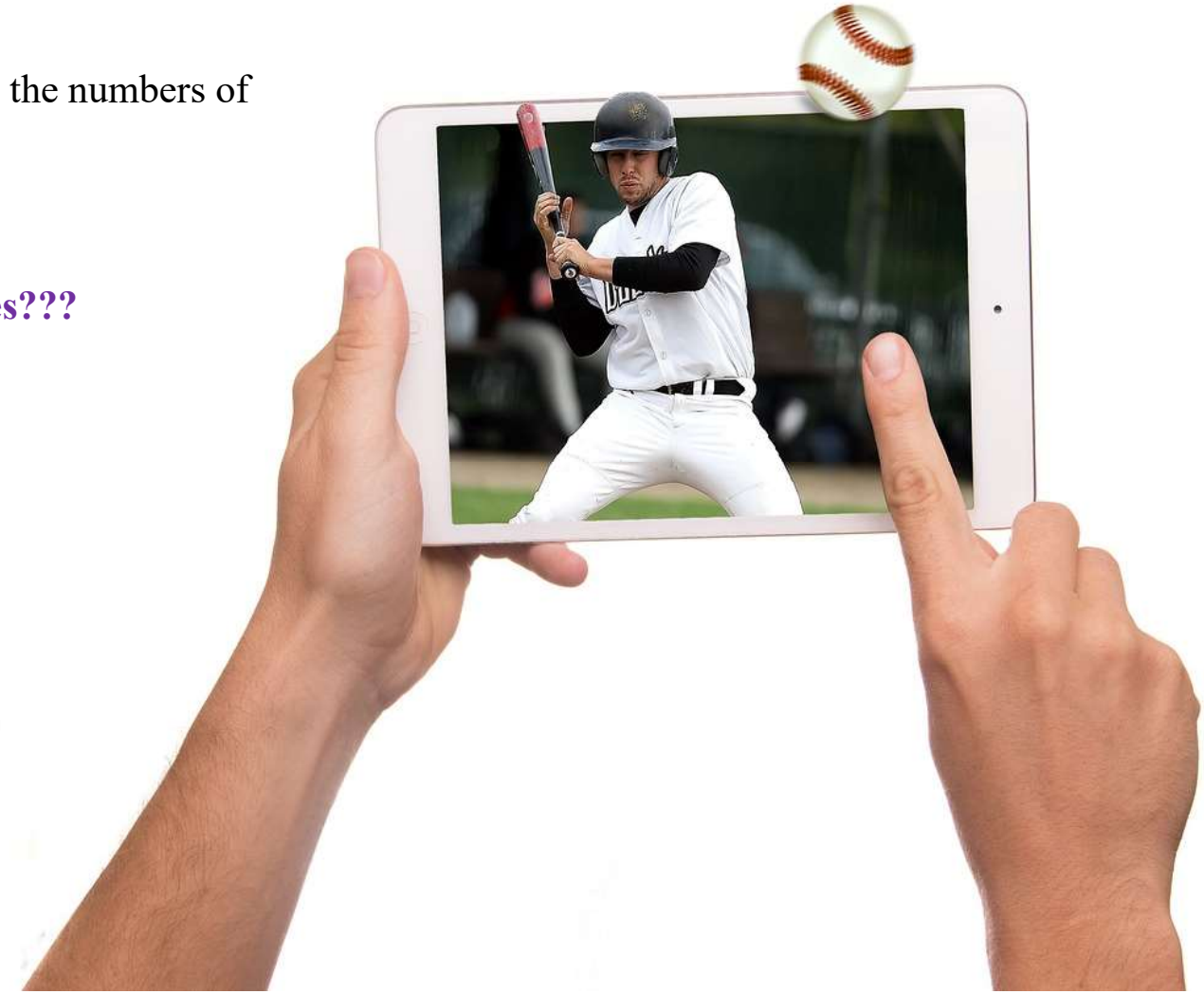
- Motivation
- Introduction to arrays
- Declaring arrays
- Examples of arrays
- 1-Dimensional array
- 2-Dimenstional array
- Passing arrays to functions
- Searching arrays
- Sorting



MOTIVATION FOR ARRAYS

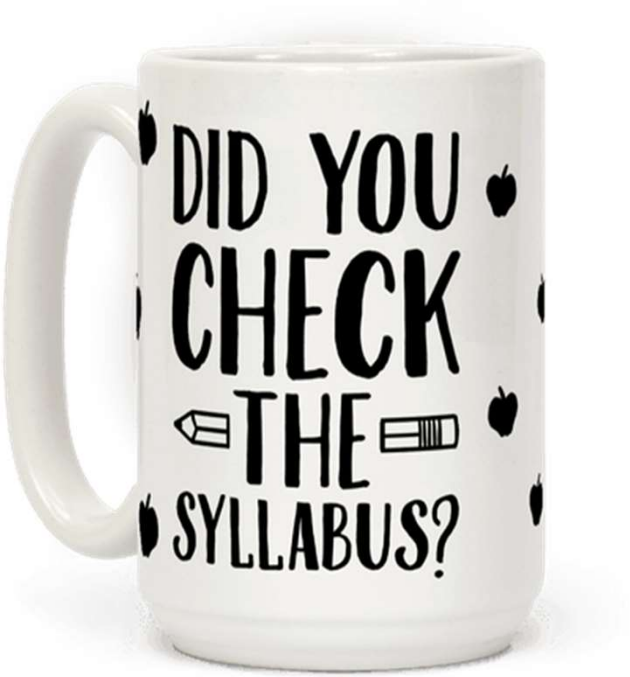
- In a program, how would you store the numbers of all the cars in a given area?
- If there are 5000 cars in an area?
- **By creating 5000 integer variables???**
- **Is it practical??**

  int a,b,c,……????



CONTENT

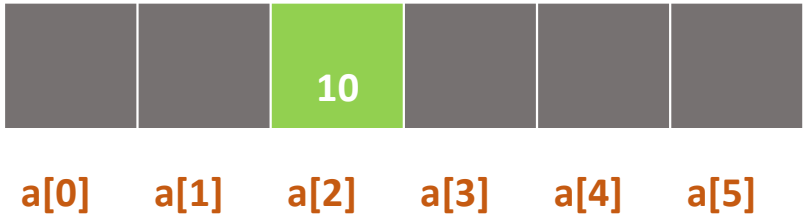
- Motivation
- **Introduction to arrays**
- Declaring arrays
- Examples of arrays
- 1-Dimensional array
- 2-Dimenstional array
- Passing arrays to functions
- Searching arrays
- Sorting arrays



INTRODUCTION TO ARRAYS

- An array is an indexed data structure to represent several variables having the same data type
- An array is represented as given below:

- An *element* of an array is accessed using the **array name** and an **index or subscript**, *for example: a[2]=10*




```
1  #include <stdio.h>
2  int main(void)
3  {
4      //Declaring an integer array
5      int a[6];
6      return 0;
7  }
```

https://www.codechef.com/ide

INTRODUCTION TO ARRAYS CONT..

- In C, the subscripts always start with 0 and increment by 1, so a[5] is the sixth element

a[0] a[1] a[2] a[3] a[4] a[5]

- An *element* of an array is accessed using the **array name** and an **index or subscript**, *for example: a[2]=220*

200	210	220	230	240	300
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

```c
#include <stdio.h>
int main()
{
    //Declaring an integer array
    int a[6];
    a[0]=200;
    a[1]=210;
    a[2]=220;
    a[3]=230;
    a[4]=240;
    a[5]=300;
    printf("%d",a[5]);
    return 0;
}
```

https://www.codechef.com/ide

DECLARATION AND INITIALISATION

- An array is defined using a declaration statement.

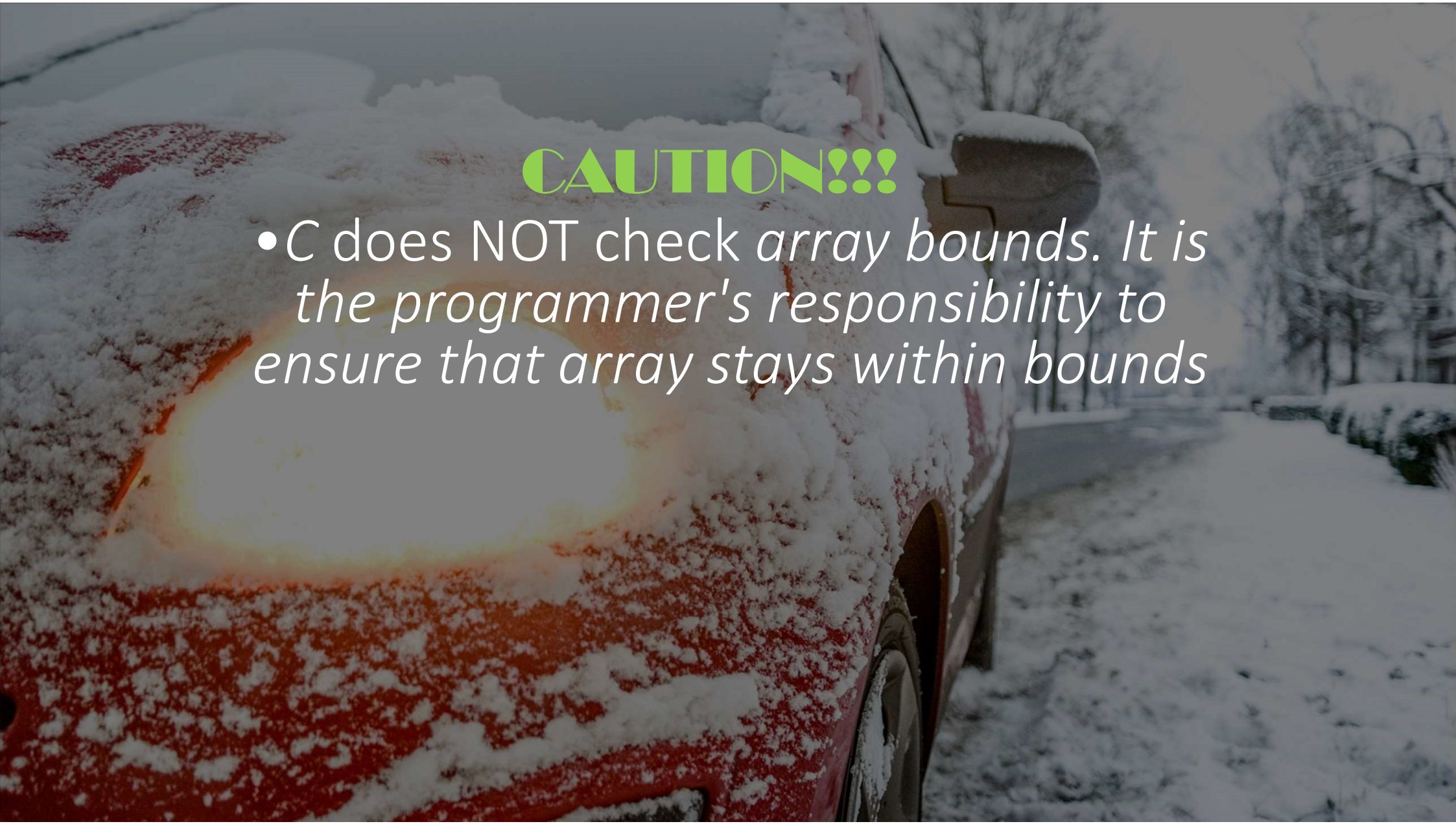
```
data_type array_name[size];
```

- allocates memory for size elements
- subscript of first element is 0
- subscript of last element is **size-1**
- **size** must be a constant

- Subscript of the first element is
  - 0
- Subscript of the last element is
  - 5

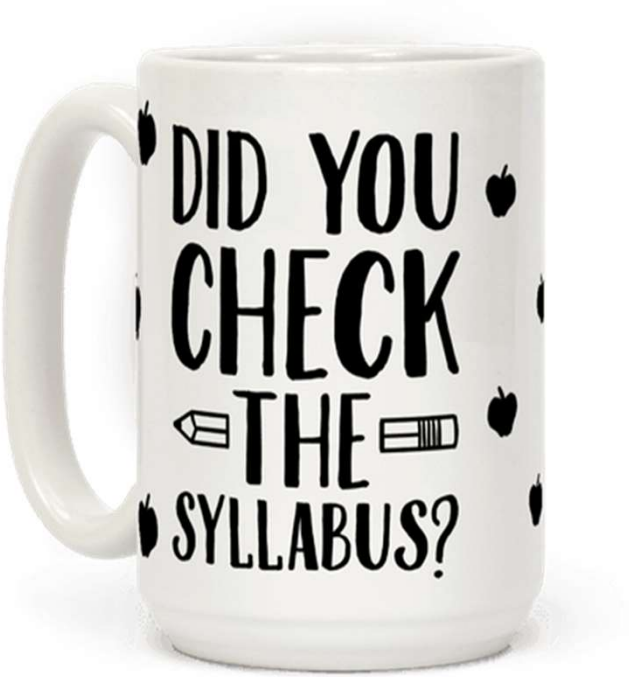
200	210	220	230	240	300
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

“C does not check bounds on arrays. Do you remember the previous slide where a[5] is the last element. Well, you know, if you give a[6] = 10, the compiler might throw a segmentation fault or may overwrite another memory location. “


CAUTION!!!
•C does NOT check array bounds. It is the programmer's responsibility to ensure that array stays within bounds

CONTENT

- Motivation
- Introduction to arrays
- **Declaring arrays**
- Examples of arrays
- 1-Dimensional array
- 2-Dimenstional array
- Passing arrays to functions
- Searching arrays
- Sorting arrays



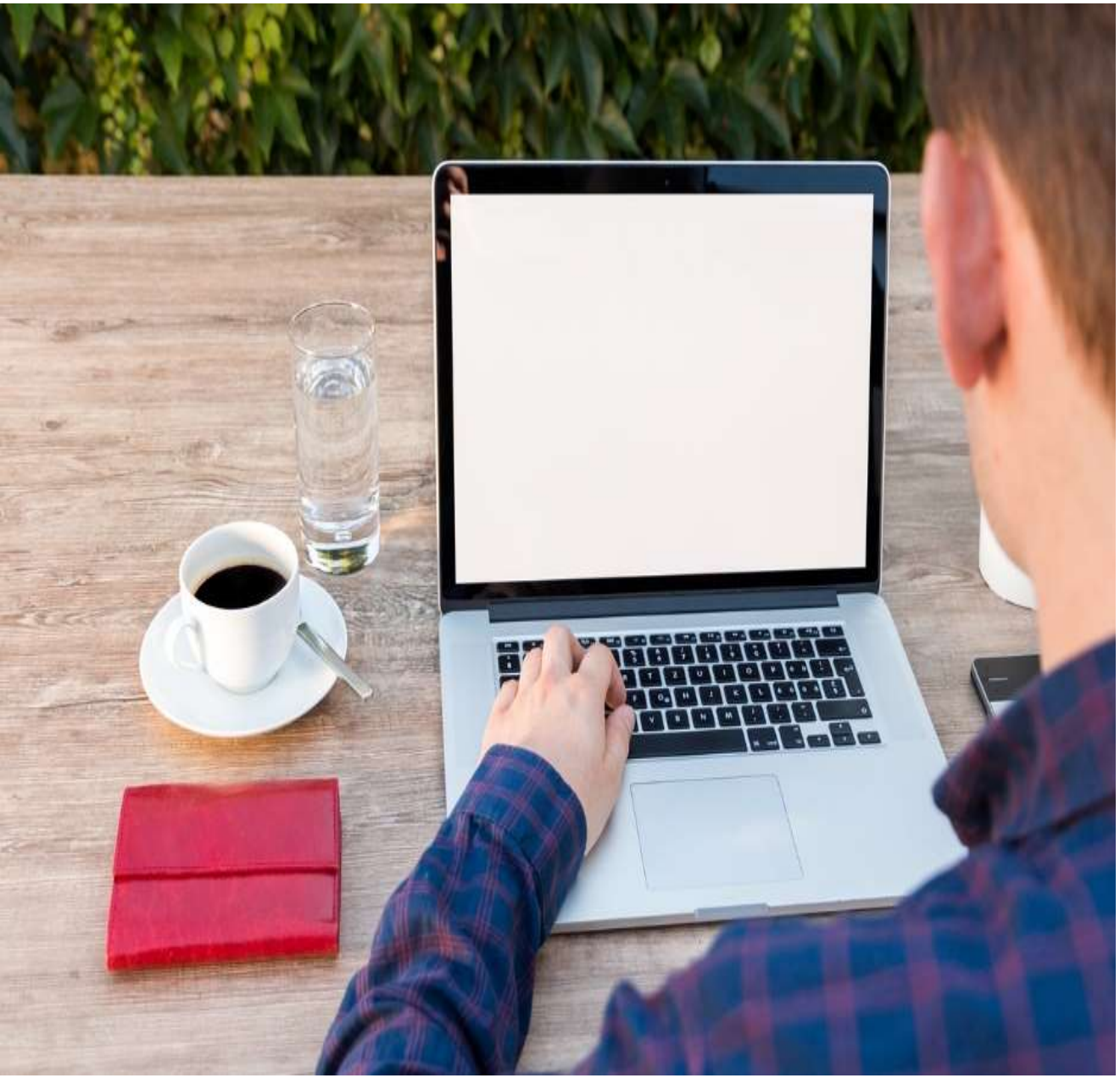
DECLARATION AND INITIALISATION CONT..

- Arrays can be initialized at the time they are declared

  - *double taxrate[3] ={0.15, 0.25, 0.3};*
  - *char list[5] = {'h', 'e', 'l', 'l', 'o'};*
  - *double vector[100] = {0.0}; /* assigns zero to all 100 elements */*
  - *int s[] = {5,0,-5}; /*the size of s is 3*/*

```
1  #include <stdio.h>
2  int main()
3  {
4      //Declaring an integer array
5      int a[6];
6      /*Assigning values to an array
7      using for loop*/
8      for(i=0;i<6;i++)
9      {
10         a[i]=i+10;
11     }
12     return 0;
13 }
```

```c
//Find_the_output_1.c
#include <stdio.h>
int main()
{
    //Declaring an integer array
    int a[6],i;
    /*Assigning values to an array
    using for loop*/
    for(i=0;i<6;i++)
    {
        a[i]=i+10;
    }
    /*Displaying the array values*/
    for(i=0;i<=6;i++)
    {
        printf("%d\n",a[i]);
    }
    /*Write your observation regarding the
    output as comments*/
    return 0;
}
```



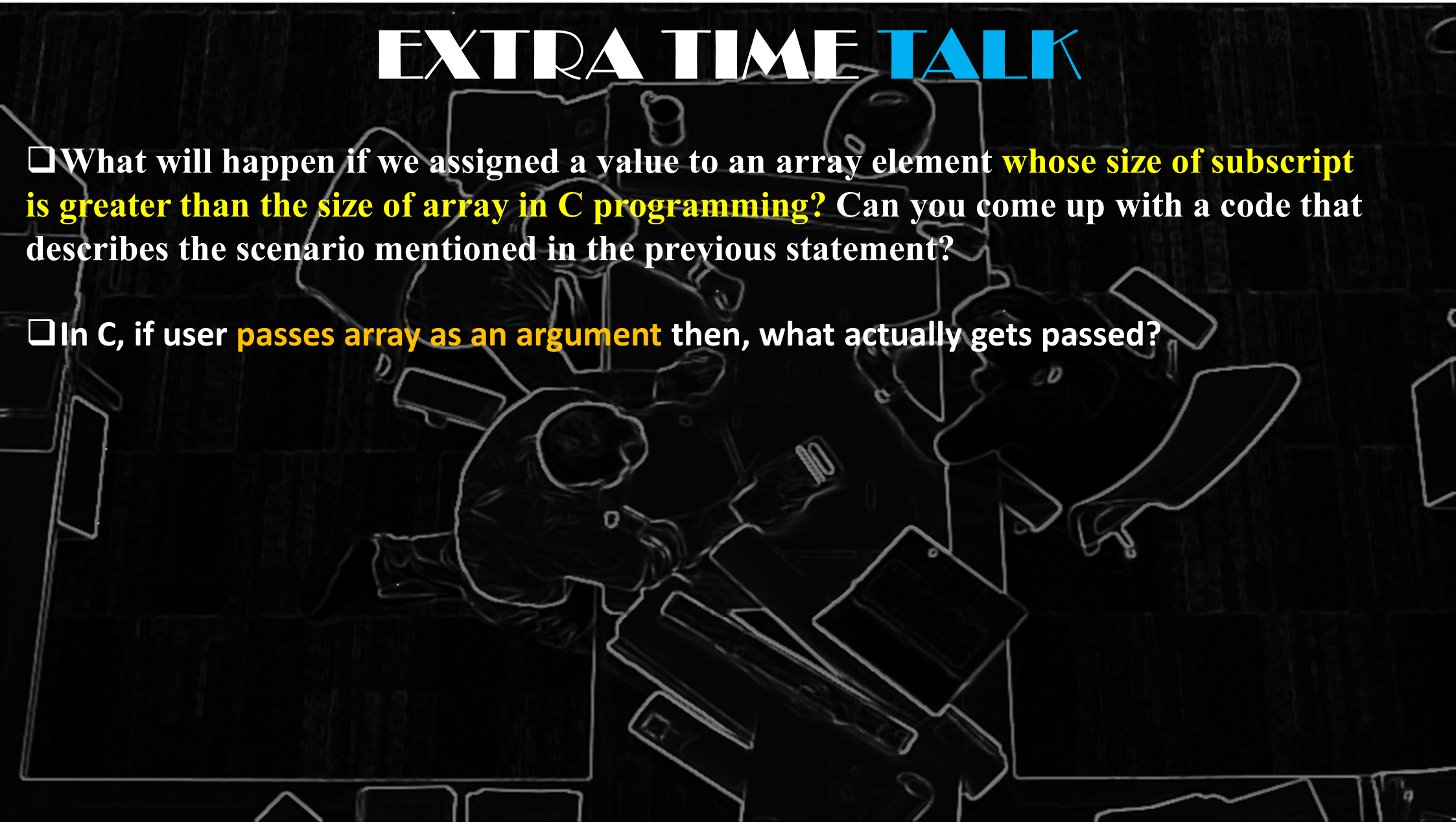
```c
//Find_the_output_2.c
#include <stdio.h>
int main()
{
    //Declaring an integer array
    char a[7]={'N','a','m','a','s','t','e'};
    printf("%d\n",a[2]);
    /*Write your observation regarding the
    output as comments*/
    return 0;
}
```

```c
//Find_the_output_3.c
#include <stdio.h>
int main()
{
    //Declaring an integer array
    char a[7]={'N','a','m','a','s','t','e'};
    printf("%d\n",a[2]+1);
    /*Write your observation regarding the
    output as comments*/
    return 0;
}
```

HOMEWORK

1. Write a program that finds the average of 100 random integer numbers in an array
2. Write a program that finds the maximum of 5 floating point numbers in an array
3. Write a program that finds the sum of every pair in a 10 integer element array and displays all the sums.

EXTRA TIME TALK



- What will happen if we assigned a value to an array element whose size of subscript is greater than the size of array in C programming? Can you come up with a code that describes the scenario mentioned in the previous statement?

- In C, if user passes array as an argument then, what actually gets passed?

“Find the output time-1”

```c
//Array-Find_output_1.c
#include <stdio.h>
int main()
{
    int arr[8];
    int n=0;
    arr[n]=++n;
    printf("%d\n%d",arr[0],arr[1]);
    return 0;
}
```

```c
//Array-Find_output_2.c
#include <stdio.h>
int main()
{
    char arr[]={ 'A', 'B', 'c', 'd', 'E', 'f' };
    int size=sizeof(arr)/sizeof(arr[0]);
    printf("%d\n", size);
    return 0;
}
```

"Find the output time-2"

```c
//Array-Find_output_3.c
#include<stdio.h>
int main()
{
    int arr[1]={10};
    printf("%d\n", arr[0]);
    return 0;
}
```

```c
//Array-Find_output_4.c
//You will have to run the code
//to find the output for this code
#include<stdio.h>
int main()
{
    int arr[] = {12, 14, 15, 23, 45};
    printf("%u, %u\n", arr, &arr);
    return 0;
}
```

“Find the output time-3”

```c
//Array-Find_output_5.c
#include<stdio.h>
int main()
{
    float arr[] = {12.4, 2.3, 4.5, 6.7};
    printf("%d\n", sizeof(arr)/sizeof(arr[0]));
    return 0;
}
```

“Find the output time-4”

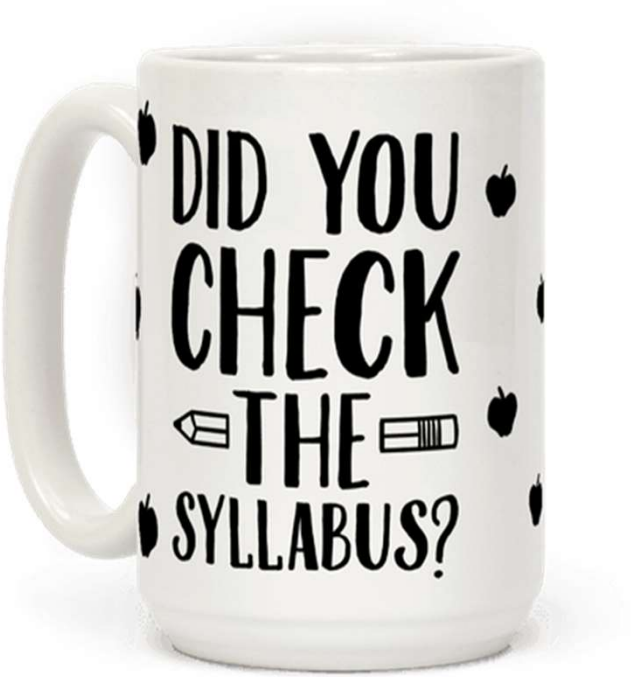
```c
#include<stdio.h>
//Use a pen and paper to answer this
//You might have to work a little
int main()
{
    int a[5] = {5, 1, 15, 20, 25};
    int i, j, m;
    i = ++a[1];
    j = a[1]++;
    m = a[i++];
    printf("%d, %d, %d", i, j, m);
    return 0;
}
```

“Find the output time - 5"

```c
#include<stdio.h>
//Use a pen and paper to answer this
//You might have to work a little
int main()
{
    int a[5] = {5, 1, 15, 20, 25};
    int i, j, m;
    i = ++a[1];
    j = a[1]++;
    m = a[i++];
    printf("%d, %d, %d", i, j, m);
    return 0;
}
```

CONTENT

- Motivation
- Introduction to arrays
- Declaring arrays
- Examples of arrays
- 1-Dimensional array
- 2-Dimenstional array
- Passing arrays to functions
- Searching arrays
- **Sorting arrays**

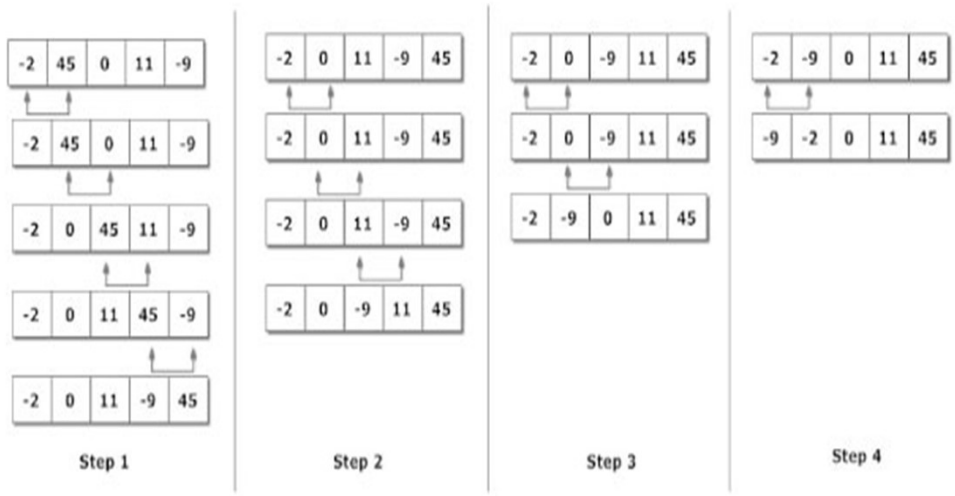


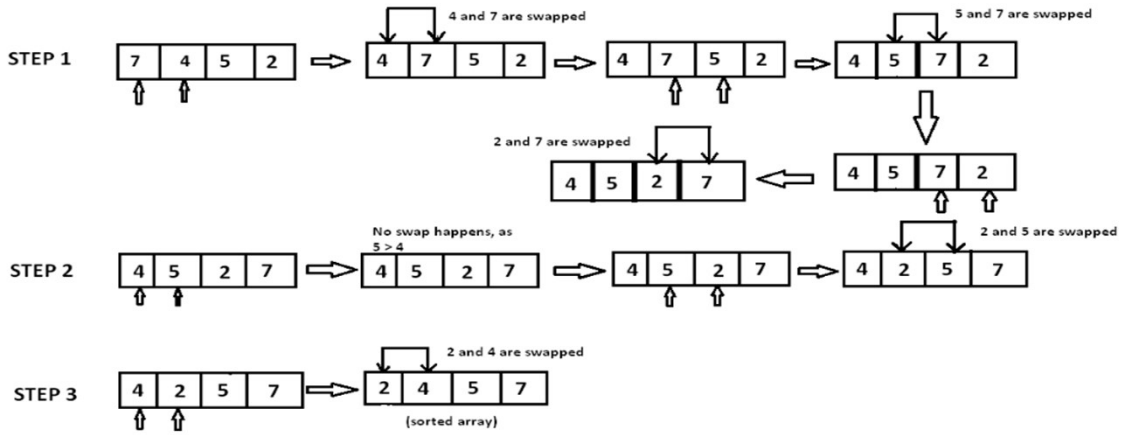
Sort the numbers in an array in ascending order

- **Bubble sort algorithm** starts by comparing the first two elements of an array and swapping if necessary, i.e., if you want to sort the elements of array in ascending order and if the first element is greater than second then, you need to swap the elements but, if the first element is smaller than second, you mustn't swap the element. Then, again second and third elements are compared and swapped if it is necessary and this process go on until last and second last element is compared and swapped. This completes the first step of bubble sort.

- If there are n elements to be sorted then, the process mentioned above should be repeated n-1 times to get required result

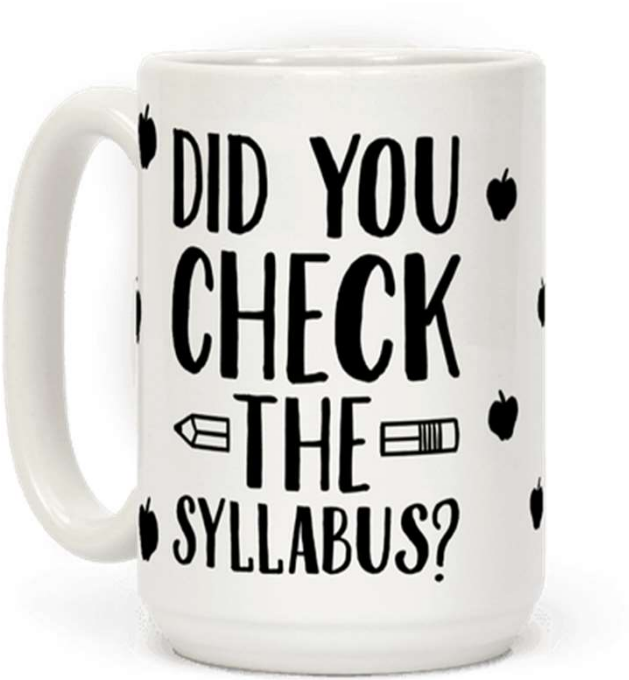
Two examples of BUBBLE SORT algorithm





CONTENT

- Motivation
- Introduction to arrays
- Declaring arrays
- Examples of arrays
- 1-Dimensional array
- **2-Dimensional array**
  - **Matrix addition, multiplication**
- Passing arrays to functions
- Searching arrays
- Sorting arrays



2d ARRAYS

- There are 3 students and each of them have received 3 marks. Write a program that can accepts all the marks of all the students.

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

	Mark 0	Mark 1	Mark 2
Student 0	40	45	50
Student 1	50	45	45
Student 2	45	45	42

- Elements in two-dimensional arrays are commonly referred by x[i][j] where i is the row number and 'j' is the column number.
- A two – dimensional array can be seen as a table with 'x' rows and 'y' columns where the row number ranges from 0 to (x-1) and column number ranges from 0 to (y-1). A two – dimensional array 'x' with 3 rows and 3 columns is shown below:

2d ARRAYS cont...

```c

#include <stdio.h>

const int STUDENT = 3;
const int MARK =3;

int main()
{
    int i=0,j=0,x[3][3];
    //Accepting user input
    for(i=0;i<STUDENT;i++)
    {
        for(j=0;j<MARK;j++)
        {
            printf("\nEnter Student %d Mark %d: ",i,j);
            scanf("%d",&x[i][j]);
        }
    }
    //Displaying the input entered by the user
    for(i=0;i<STUDENT;i++)
    {
        for(j=0;j<MARK;j++)
        {
            printf("\nStudent %d  Mark %d: is %d",i,j,x[i][j]);
            scanf("%d",&x[i][j]);
        }
    }
    return 0;
}
```

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

	Mark 0	Mark 1	Mark 2
Student 0	40	45	50
Student 1	50	45	45
Student 2	45	45	42

2d ARRAYS cont...

Complete the code that finds out the sum of the main diagonal of the matrix

40	45	50
50	45	45
45	45	42

40	45	50
50	45	45
45	45	42

Why? Explain the reason

```c
/**
 * C program to find sum of main diagonal elements of a matrix
 */

#include <stdio.h>

#define SIZE 3 // Matrix size

int main()
{
    int A[SIZE][SIZE];
    int row, col, sum = 0;

    /* Input elements in matrix from user */
    printf("Enter elements in matrix of size %dx%d: \n", SIZE, SIZE);
    for(row=0; row<SIZE; row++)
    {
        for(col=0; col<SIZE; col++)
        {
            scanf("%d", &A[row][col]);
        }
    }

    /* Find sum of main diagonal elements */
    for(     ;       ;       )
    {
        sum =     + A[   ][   ];
    }

    printf("\nSum of main diagonal elements = %d", sum);

    return 0;
}
```

2d ARRAYS cont...Matrix Multiplication

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \times \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{pmatrix}$$

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}}_{1\times 3} \cdot \underbrace{\begin{bmatrix} 2 & 1 & 3 \\ 3 & 3 & 2 \\ 4 & 1 & 2 \end{bmatrix}}_{3\times 3} = \underbrace{\begin{bmatrix} 1\cdot 2+2\cdot 3+3\cdot 4 \\ 1\cdot 1+2\cdot 3+3\cdot 1 \\ 1\cdot 3+2\cdot 2+3\cdot 2 \end{bmatrix}}_{1\times 3} = \begin{bmatrix} 20 \\ 10 \\ 13 \end{bmatrix}$$

A matrix multiplication reminder

mxn

	00	01
00	1	2
10	3	4

pxq

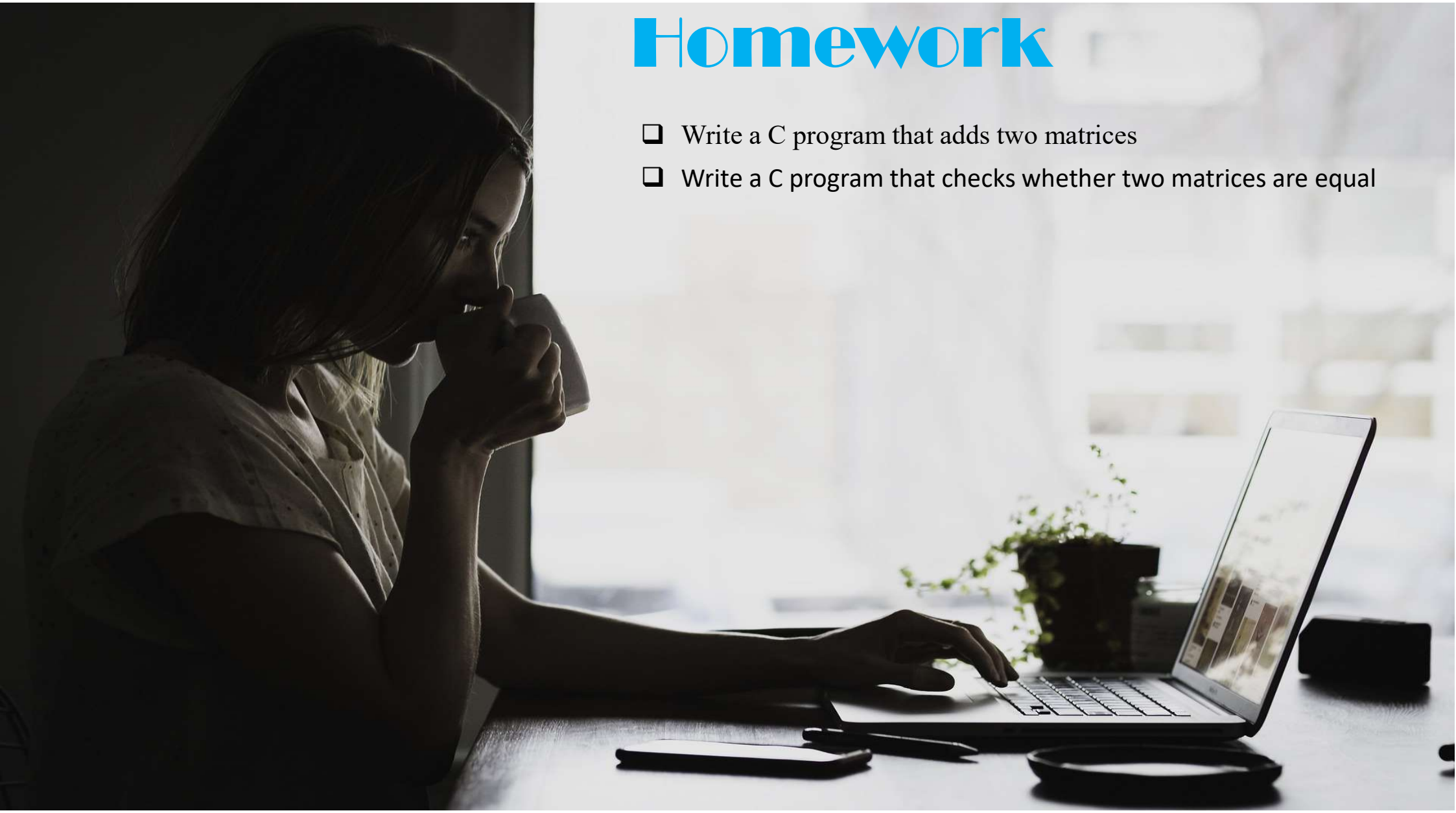
	00	01	02
00	10	20	50
10	30	40	60

```
for (c = 0; c < m; c++) {
  for (d = 0; d < q; d++) {
    for (k = 0; k < p; k++) {
      sum = sum + first[c][k]*second[k][d];
    }
```

c	d	k
0	0	0
0	0	1

first	second	Value
1	10	10
2	10	20

first	second	Value
1	10	10
2	30	60

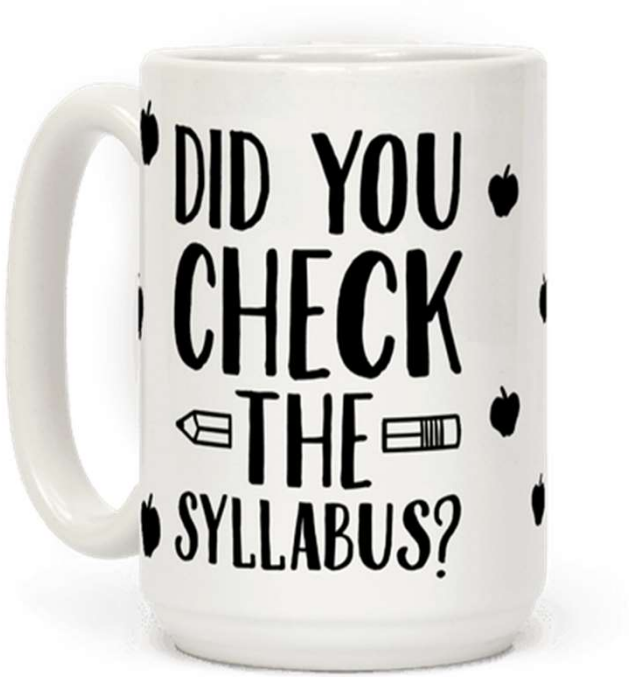


Homework

- Write a C program that adds two matrices
- Write a C program that checks whether two matrices are equal

CONTENT

- Motivation
- Introduction to arrays
- Declaring arrays
- Examples of arrays
- 1-Dimensional array
  - Strings
- 2-Dimensional array
- Passing arrays to functions
- Searching arrays
- Sorting arrays



STRINGS IN C

Write a program that reads a name from the user and displays it

Character 1D Array

```c
#include <stdio.h>
int main()
{
   char str[20];
   scanf("%s",str);
   printf("%s", str);
   return 0;
}
```

```c
#include <stdio.h>
int main()
{
   char str[20];
   gets(str);
   printf("%s", str);
   return 0;
}
```

Give the input as: Good Afternoon

Output would be: Good

What's the difference

Give the input as: Good Afternoon

Output would be: Good Afternoon

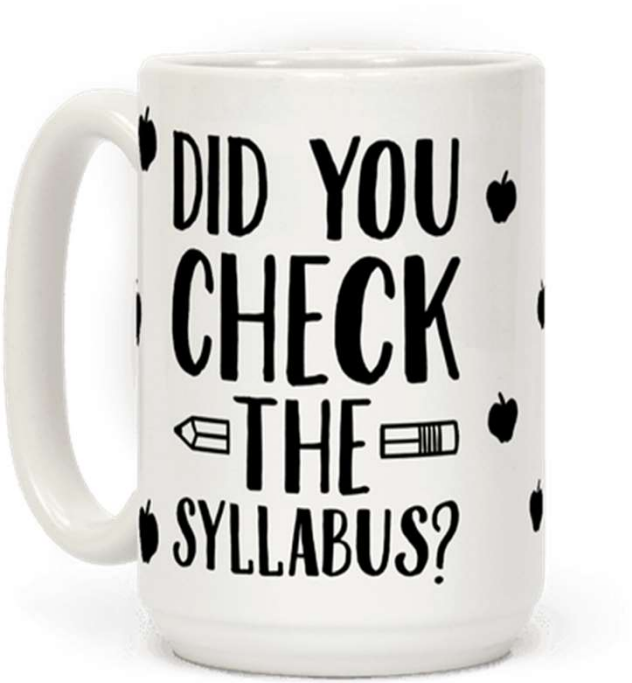

'gets' is
WHY

Write a program that reads a name from the user and displays it [without gets]

```c
#include <stdio.h>
#define MAX_LIMIT 4
int main()
{
char str[MAX_LIMIT];
fgets(str, MAX_LIMIT, stdin);
printf("%s", str);
return 0;
}
```

CONTENT

- Motivation
- Introduction to arrays
- Declaring arrays
- Examples of arrays
- 1-Dimensional array
  - Strings
    - strcmp(), strlen(), strcat(), strcpy()
- 2-Dimensional array
- Passing arrays to functions
- Searching arrays
- Sorting arrays

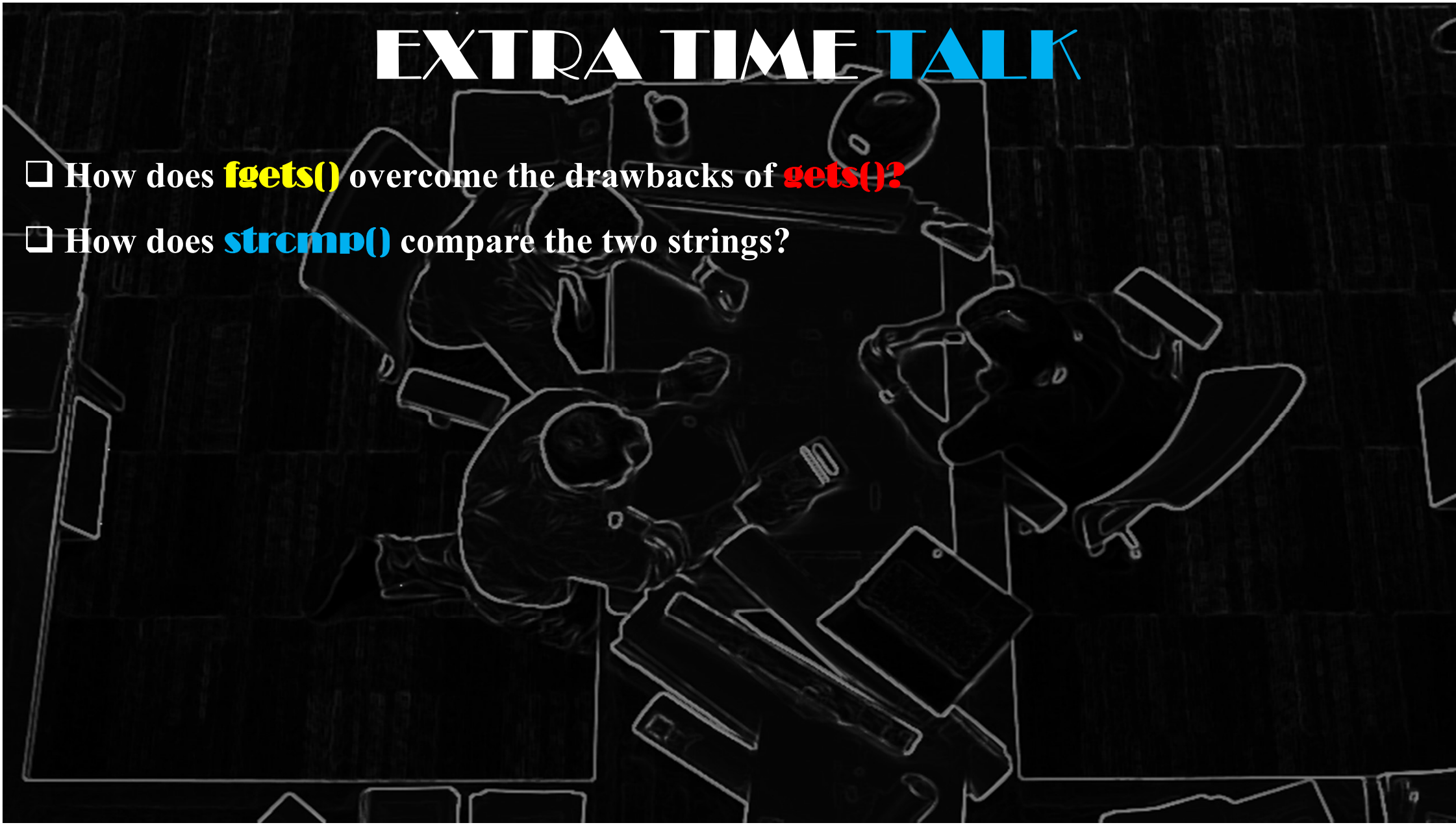


STRING comparison using strcmp

```c
#include<stdio.h>
#include<string.h>

#define MAX_STRING_LEN 80

int main()
{
 int res=0;

 char str[MAX_STRING_LEN];
 char str1[MAX_STRING_LEN];

 printf("Enter the first string: ");
 fgets(str, MAX_STRING_LEN,stdin);
 printf("\nEnter the second string: ");
 fgets(str1,, MAX_STRING_LEN,stdin);

 /*strcmp() is a built-in function
 in the string.h file
 */
 res = strcmp(S1,S2);
 if(res==0)
 {
    printf("The strings are equal");
 }
   return 0;
}
```

- **strcmp()** is a built-in library function and is declared in **<string.h>** header file.
- This function takes two strings as arguments and compare these two strings **lexicographically.**

EXTRA TIME TALK



- How does **fgets()** overcome the drawbacks of **gets()**?
- How does **strcmp()** compare the two strings?

strcmp() – Find the output 2 – Give reason

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char str1[] = "abcd", str2[] = "abCd", str3[] = "abcd";
    int result;

    // comparing strings str1 and str2
    result = strcmp(str1, str2);
    printf("strcmp(str1, str2) = %d\n", result);

    // comparing strings str1 and str3
    result = strcmp(str1, str3);
    printf("strcmp(str1, str3) = %d\n", result);

    return 0;
}
```

Find the output 3 – **Give reason**

```c
//C program to calculate the length of string

#include <stdio.h>
int main()
{
   char s1[] = "Namaste";
   int i = 0;
   printf("The length of the string is %d",strlen(s1));
   return 0;
}
```

Complete the code 1 – **Give reason**

```
1
2  /* Program to find the
3  number of times a particular Character
4  appears in a given string */
5
6  #include <stdio.h>
7
8  int main()
9  {
10    char s[] = "Baudhayana";      // String Given
11    char ch = 'a';               // Character to count
12
13    int i = 0;
14    int count = 0;               // Counter
15
16    while(____ != '\0')
17    {
18      if(____ == ch)
19        count++;
20      i++;
21    }
22
23    printf("The character %c appears %d times",ch,count);
24    return 0;
25 }
```

Write the code 1

Write the code to count the number of vowels in a given string

strcat() – find the output code 1

```c
#include <stdio.h>
#include <string.h>

int main( )
{
   char str1[ ] = "Best way of learning" ;
   char str2[] = " is learning by doing" ;

   strcat ( str1, str2 ) ;

   printf ( "after strcat( ) = %s",str1 ) ;
   return 0;
}
```

Implement your own strcat() and also study strcpy()

Find the output

```c
#include<stdio.h>
#include<string.h>

int main()
{
    char str1[20] = "Hello", str2[20] = " World";
    printf("%s\n", strcpy(str2, strcat(str1, str2)));
    return 0;
}
```

Find the output


```c
#include<stdio.h>

int main()
{
    char p[] = "%d\n";
    p[1] = 'c';
    printf(p, 65);
    return 0;
}
```

Pic by Joshua Sortino

Find the output

Find the output

```c
#include<stdio.h>
#include<string.h>

int main()
{
    printf("%d\n", strlen("123456"));
    return 0;
}
```

Find the output

This is a difficult one!!!

```c
#include<stdio.h>
int main()
{
    printf(5+"Good Morning\n");
    return 0;
}
```

Find the output

(Think a while)

```c
#include<stdio.h>
#include<string.h>

int main()
{
    char str[] = "India\0\BIX\0";
    printf("%s\n", str);
    return 0;
}
```

Pic by Donald Teel

Find the output – Better understanding

FUNCTIONS

```
#include<stdio.h>
int main()
{
    fun();
    printf("\n");
    return 0;
}
fun()
{
    char c;
    if((c = getchar())!= '\n
        fun();
    printf("%c", c);
}
```

Find the output – Better understanding

STRING

```c
#include<stdio.h>
int main()
{
    // 8 characters in a string of SIZE 7!!
    char str[7] = "IndiaWON";
    printf("%c\n", str[6]);
    return 0;
}
```

Passing Array to a Function

- There are two ways of passing data to a function
  - Pass by VALUE
  - Pass by REFERENCE
- ARRAYS are always **passed by REFERENCE**

```
int main()
{
    int values[5], i, max;

    printf("Enter 5 numbers\n");
    for( i = 0; i < 5; ++i )
        scanf("%d", &values[i] );
    //The entire array is passed to the function
    // called 'maximum'
    max = maximum( values );
    printf("\nMaximum value is %d\n", max );
    return 0;
}
```

Passing Array to a Function

- There are two ways of passing data to a function
  - Pass by VALUE
  - Pass by REFERENCE
- ARRAYS are always **passed by REFERENCE**

```
#include <stdio.h>
//Function prototype
int maximum( int [] );
int  maximum( int arr[5] )
{
    int  max_value, i;

    max_value = arr[0];
    for( i = 0; i < 5; ++i )
        if( arr[i] > max_value )
            max_value = arr[i];

    return max_value;
}
```

```c
#include <stdio.h>
//Function prototype
int maximum( int [] );
int  maximum( int arr[5] )
{
    int  max_value, i;

    max_value = arr[0];
    for( i = 0; i < 5; ++i )
        if( arr[i] > max_value )
            max_value = arr[i];

    return max_value;
}
```

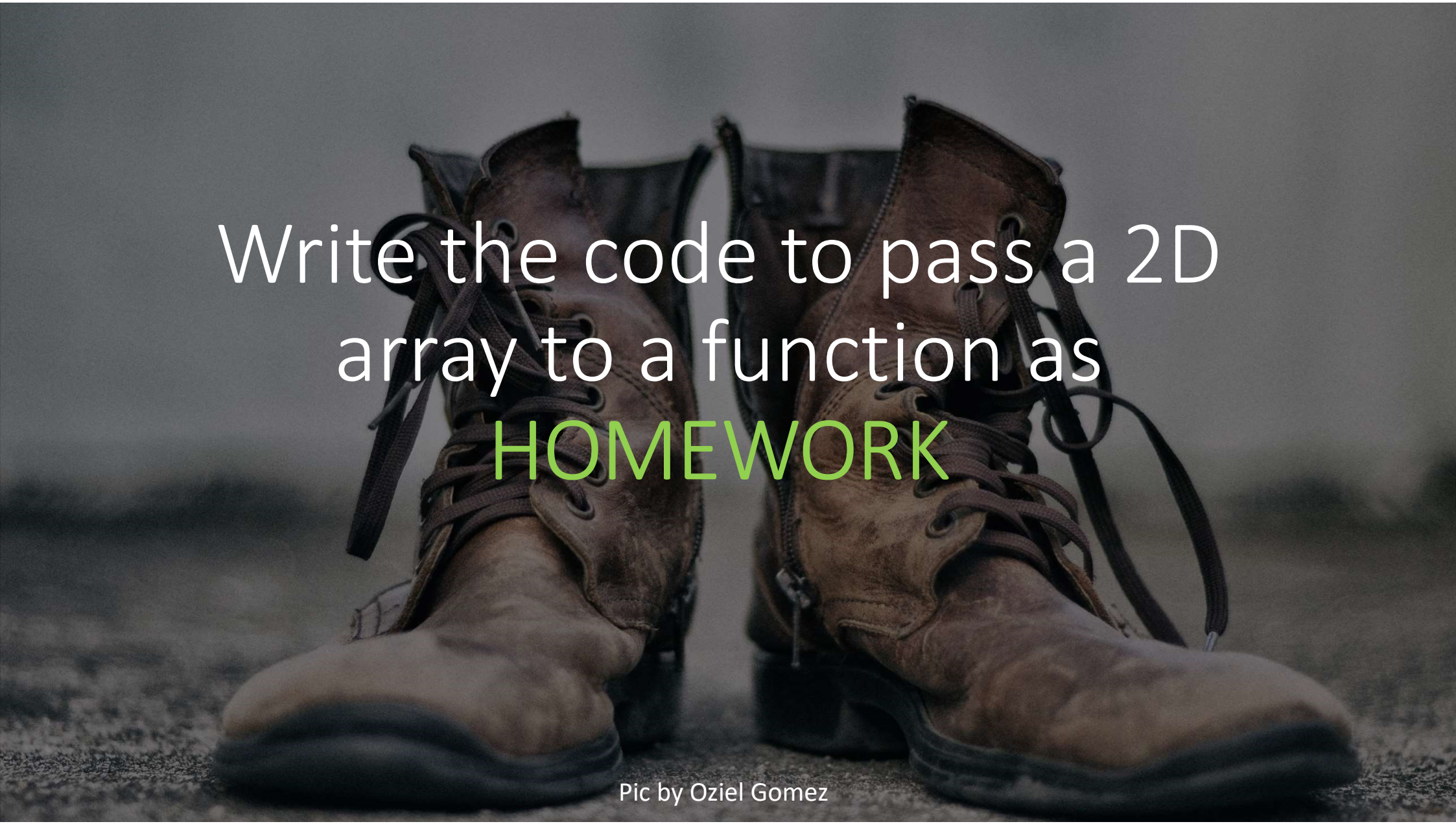
```c
int main()
{
    int values[5], i, max;

    printf("Enter 5 numbers\n");
    for( i = 0; i < 5; ++i )
        scanf("%d", &values[i] );
    //The entire array is passed to the function
    // called 'maximum'
    max = maximum( values );
    printf("\nMaximum value is %d\n", max );
    return 0;
}
```

```c
#include <stdio.h>
void display(int n[])
{
    //The format specifier is 'd'
    //notice the output
    printf("%d",n);
}

int main()
{
    int num[] = { 2, 3, 4 };
    display(num); //Passing the entire array
    return 0;
}
```

```c
#include <stdio.h>
void display(int n[])
{
    //The format specifier is 'u'
    //notice the output
    printf("%u",n);
}

int main()
{
    int num[] = { 2, 3, 4 };
    display(num); //Passing the entire array
    return 0;
}
```

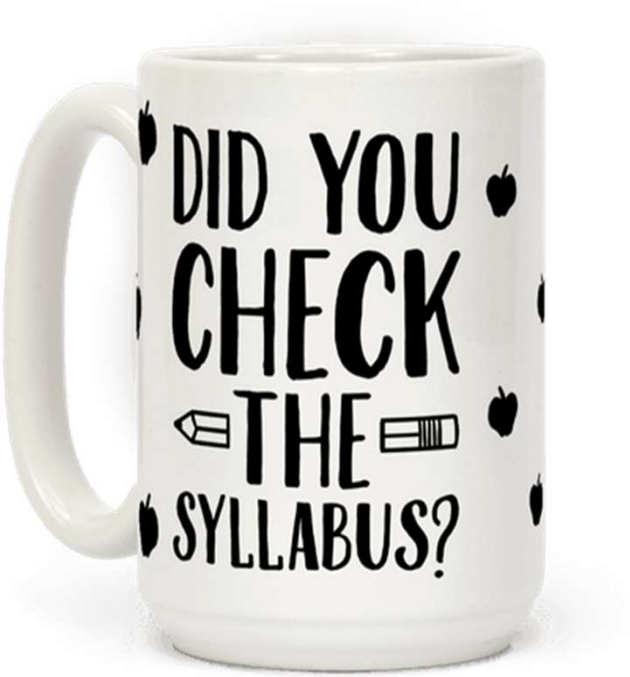
```c
#include <stdio.h>
void display(int n[])
{
    //Study the output of the following
    //code
    printf("%u\n%u\n%u",n,&n,&n[0]);
}

int main()
{
    int num[] = { 2, 3, 4 };
    display(num); //Passing the entire array
    return 0;
}
```

Study anything new??


Write the code to pass a 2D array to a function as
HOMEWORK
Pic by Oziel Gomez

CONTENT

- Motivation
- Introduction to arrays
- Declaring arrays
- Examples of arrays
- 1-Dimensional array
- 2-Dimensional array
- Passing arrays to functions
- **Pointers**
- Searching arrays
- Sorting arrays



Pointers

```c
#include <stdio.h>
int main()
{
    int *ptr, q;
    q = 50;
    /* address of q is assigned to ptr */
    ptr = &q;
    /* display q's value using ptr variable */
    printf("%d", *ptr);
    return 0;
}
```

```c
#include<stdio.h>
void xyz(int);
int main()
{
    int a = 10;
    printf("The address of a is %u\n",&a);
    /*

    Remember, when you are passing via
     'pass by value' you are passing a
     a copy of the value, in this case
     value of 'a'
    xyz(a);

    */

    return 0;
}
void xyz(int m)
{
    printf("The value of a is %d\n",m);
    printf("The address of m is %u",&m);
}
```

OUTPUT

```
The address of a is 2216612116
The value of m is 10
The address of m is 2216612092
```

```c
#include<stdio.h>
void xyz(int);
int main()
{
    int a = 10;
    printf("The address of a is %u\n",&a);
    /*

    Remember, when you are passing via
     'pass by value' you are passing a
     a copy of the value, in this case
     value of 'a'
    xyz(a);

    */
    xyz(a);
    printf("In main function again \n The value of a is %d\n",a);
    return 0;
}
void xyz(int m)
{
    m = m+10;
    printf("The value of m is %d\n",m);
    printf("The address of m is %u\n",&m);
}
```

Just to re-inforce the idea that ONLY the copy of the value is being passed to the function 'xyz'

a=10

m=10

'Pass by reference' and POINTERS

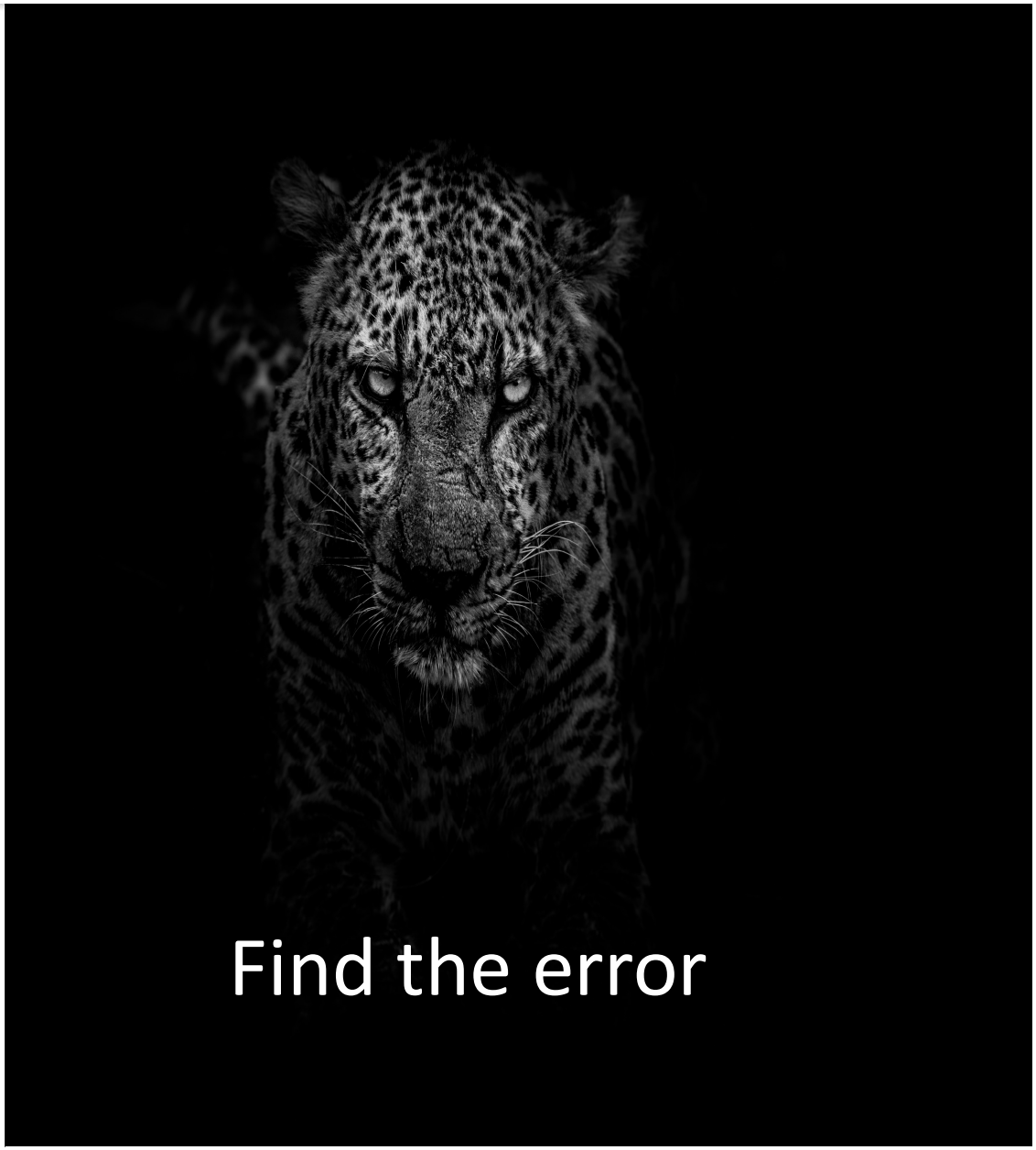
```c
#include<stdio.h>
//Change 1
void xyz(int*);
int main()
{
    int a = 10;
    printf("The address of a is %u\n",&a);
    /*

    'pass by argument - Passing the address of 'a'
    to 'xyz'

    */
    //Change 2
    xyz(&a);
    printf("In main function again \n The value of a is %d\n",a);
    return 0;
}
//Change 3
void xyz(int *m)
{
    //Change 4
    *m = *m+10;
    printf("The value of m is %d\n",*m);
    //Change 5
    printf("The address of m is %u\n",&(*m));
}
```

321080036

321080036


Find and write the difference between 'pass by value' and 'pass by reference' using 'pointers'
Pic by Forest Cavale

```c
#include<stdio.h>
void xyz(int*);
int main()
{
    int a = 10;
    xyz(a);
    return 0;
}

void xyz(int *m)
{
    *m = *m+10;
    printf("The value of m is %d\n",*m);
    printf("The address of m is %u\n",&(*m));
}

//A common error
```



```c
#include<stdio.h>
void xyz(int*);
int main()
{
    int a = 10;
    printf("The value of a before the function is called is %d\n",a);
    xyz(&a);
    printf("The value of a is after the function is called is %d\n",a);
    return 0;
}

void xyz(int *m)
{
    *m = *m+10;
    //Notice that no value is returned
}
```

Find the output

```c
int main()
{
    int num1, num2;

    printf("\nEnter the first number : ");
    scanf("%d", &num1);
    printf("\nEnter the Second number : ");
    scanf("%d", &num2);

    //Sending the address of numbers
    //to the function
    swap(&num1, &num2);

    //We get the swapped values even though
    // the function does not return the
    // swapped values

    printf("\nFirst number  : %d", num1);
    printf("\nSecond number : %d", num2);

    return (0);
}
```

```c
#include<stdio.h>

void swap(int *num1, int *num2)
{
    //*num1 will let you have the value
    // at the corresponding address
    int temp;
    temp = *num1;
    *num1 = *num2;
    *num2 = temp;
}
```

A program to SWAP two variables using POINTERS