**GP MUTHU THUNAI**

**NOTE:**

**\*Paathi questions ku thaa answers iuruku…**

**\*Eluthurathuku munaadi Check panitu eluthikonga…**

**Programming in C**

**CIE-2 Question Bank Answers**

**PART-A**

1. What is an array? How can you assign the string value “COMPUTER” in a char array variable?Array: An array in C is a collection of items (all of same data type, accessed using a common name) stored at contiguous memory locations and elements can be accessed randomly using indices of an array.

**EXAMPLE:**

#include<stdio.h>   
int main()   
{

char a[10]="COMPUTER";

a[]="COMPUTER";  
 printf("%s",a);   
 return 0;   
}

**SYNTAX:**

#include<stdio.h>  
int main()  
{  
char

printf("%s",a);  
return 0;

}

# 

# 2. Write a C program to demonstrate that the array elements are stored in contiguous locations.

#include<stdio.h>

void main()

{  
 int arr[5] = {1,1,2,3,5};

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

{

printf("\nAddress of a %d = %u", i, (arr+i) );

}

}

Output:

Address of a0 = 3908053440  
Address of a1 = 3908053444

Address of a2 = 3908053448

Address of a3 = 3908053452

Address of a4 = 3908053456

Thus, they are stored in contiguous location.

# 3. What does the following declaration mean?

**ptr is a pointer to an array of integers.**  
int (\*ptr)[10];

# What does the output of the below code? int x[10]={1,2,3,4,5,6};

# int \*y;

# y=&x[0];

# printf("%d", \*(y+4));

**Output:  
 5**

# 4. What is recursion?

A function called by itself is called recursive function. The recursive functions should be used very carefully because, when a function called by itself it enters into the in\_nite loop. And when a function enters into the in\_nite loop, the function execution never gets completed. We should de\_ne the condition to exit from the function call so that the recursive function gets terminated. When a function is called by itself, the \_rst call remains under execution till the last call gets invoked. Every time when a function  
call is invoked, the function returns the execution control to the previous function call.

# 5. Consider the character variable has the given assignment like,

# char c[10]={'C','O','R','O','N','A','!'};

# printf("%d",?);

**ANS:**

Method: strlen(c)

Output:7

# 7. What is the use of free() in dynamic memory allocation?

Dynamically allocated memory created with eithercalloc() ormalloc() doesn't get freed on their own.  
You must explicitly usefree() to release the space.

**Syntax: free(ptr);**

# 8. Consider the following code and compute the value of p+1

# main()

# {

# int a, \*p; // declaring the variable and pointer

# p=&a;

# printf("%d", p+2);

# return 0;

# }

# If the address of the variable ‘a’ is 790558580. Then, find p+1.

Value of p+1 :790558584

Value of p+2 :790558588

# 9. Execute the below code

# int main()

# {

# int a=90; // declaring the variable and pointer

# printf("%d\n", \*(&a));

# return 0;

# }

**Output:90**

# 10. Distinguish between calloc() and malloc().

**Malloc()** function will create a single block of memory of size specified by **the** user. **Calloc()** function can assign multiple blocks of memory for a variable.

**Malloc** function contains garbage value. **The** memory block allocated by a **calloc** function is always initialized to zero.

# 11. What is the output of the below code

# int main()

# {

# int a, \*p; // declaring the variable and pointer

# a = 17;

# p = &a; // initializing the pointer

# printf("%d %p", \*p,\*p);

# }

**OUTPUT:** 1700000011

# 12. Consider the following code and compute the value of p+1

# main()

# {

# int a, \*p; // declaring the variable and pointer

# p=&a;

# printf("%d", p+2);

# return 0;

# }

# If the address of the variable ‘a’ is 790558580. Then, find p+1.

**Value of p+1 :790558584**

**Value of p+2 :79055858**

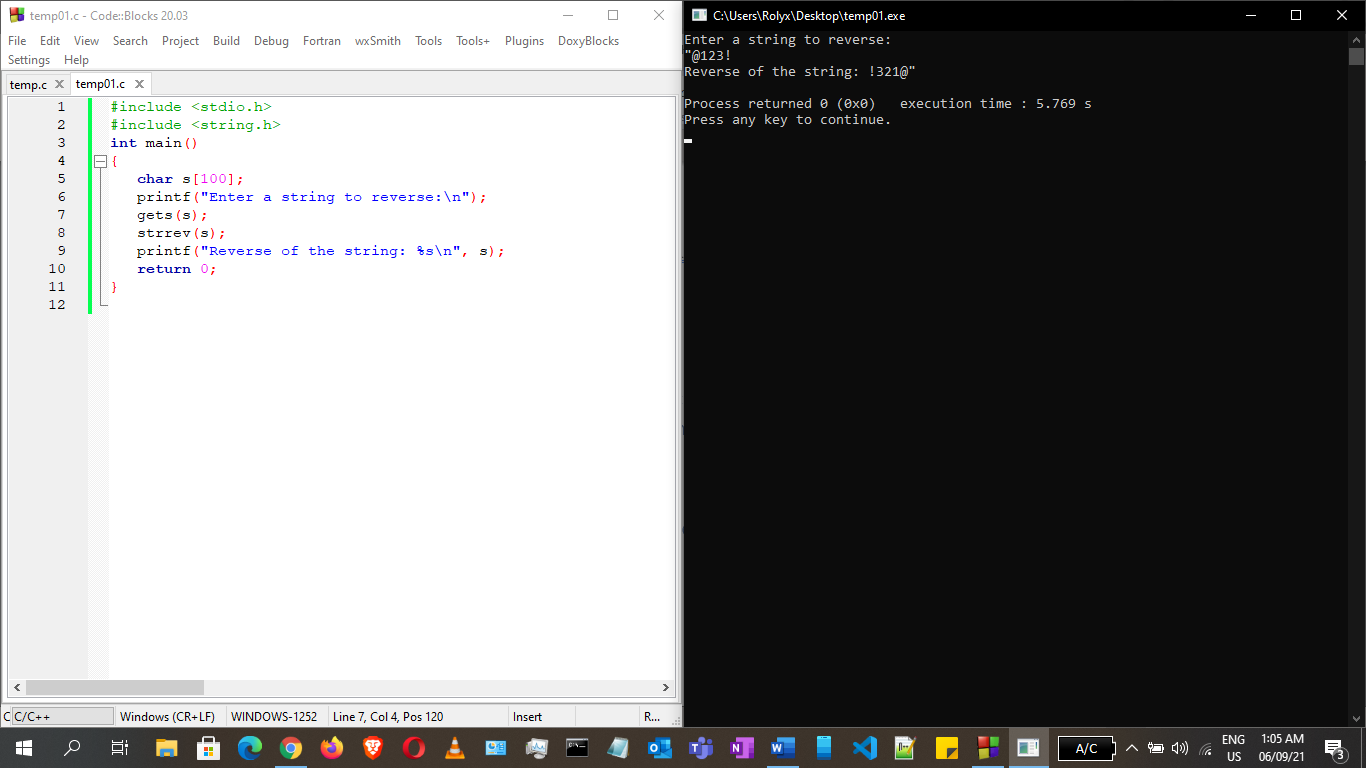
**PART-B**

# 1. Given two matrices, write a program to compute the subtraction of those matrices.

# The First matrix is : 5 6 7 8 The Second matrix is : 1 2 3 4 The Subtraction of two matrix is : ? ? ? ?

# The Multiplication of two matrices is : ? ? ? ?

# 2. Write the program for the given concept using function. If a given input is "@123!," then your function should return "!321@".



# 3. Write the program to illustrate the given two String methods with suitable example.

# **a) strcpy()**

# **b) strncpy()**

# **strcpy()**

char\*strcpy (char\*str1,char\*str2)  
It copies the string str2 into string str1, including the end  
character (terminator char ‘\0’).

**Example of strcpy:**

#include <stdio.h>

#include <string.h>

int main()

{

char s1[30] = "Time";

char s2[30] = "Is gold";

strcpy(s1,s2);

printf("S1 is: %s", s1);

return 0;

}

**Output:**

S1 is: Is gold

**strncpy()**

char \*strncpy( char \*str1, char \*str2, size\_t n)  
size\_t is unassigned short and n is a number.  
**Case1:** If length of str2 > n then it just copies first n characters of str2 into str1.  
**Case2:** If length of str2 < n then it copies all the characters of str2 into str1  
and appends several terminator chars(‘\0’) to accumulate the length of str1 to  
make it n.

**Example of strncpy:**

#include <stdio.h>

#include <string.h>

int main()

{

char s1[30] = "All";

char s2[30] = "Is Well";

strncpy(s1,s2, 12);

printf("S1 is: %s", s1);

return 0;

}

**Output:**

S1 is: Is Well

**PART-C**

# 1. Consider Antony reads five integer values from the user as 12,21,35,64,87 using array. He wants to know the position of 87. Which algorithm may helps him to finds its position and how? Write the code for it.

# 2. How to use the format strings %d, %u and %p in allocating memory addresses to a variable. Illustrate each format pointer with suitable code.

# Note: Use Pointer arithmetic

# 3. Write a program in C to find the largest element using Dynamic Memory Allocation. Consider the following sample Test Set for the execution.

# **Test Data :**

# Input total number of elements(1 to 100): 5

# Number 1: 5

# Number 2: 7

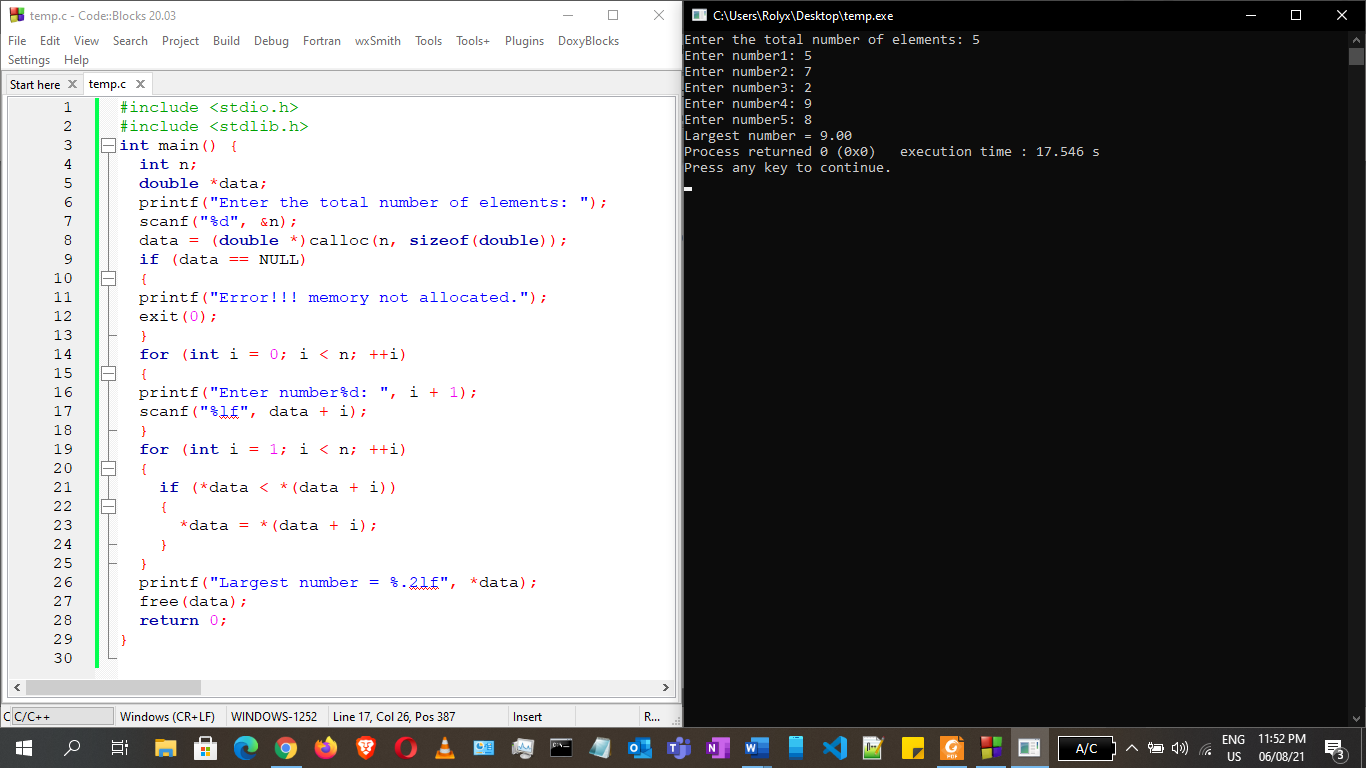
# Number 3: 2

# Number 4: 9

# Number 5: 8

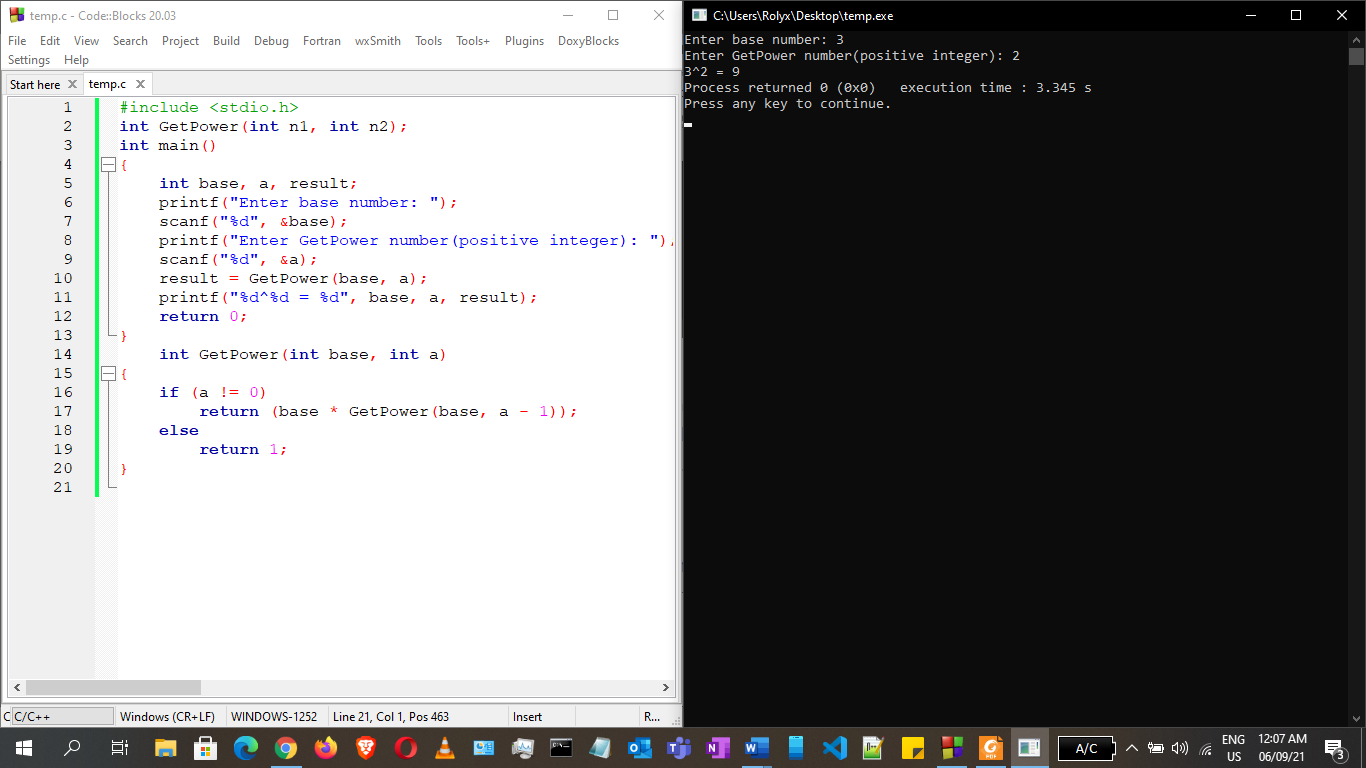
# **Expected Output :**

# The Largest element is : 9.00



# 4.Consider you have declared and assigned an array variable called Arr1={1,2,2,3,3}; How can you find the unique element present in the given Arr1. Write the program for it.

# 5. Write the c program for finding the power of any given number using function recursive. Use the function GetPower() with two arguments ‘base’ and ‘power’



# 6. Write a C program to sort the below mentioned array of elements using Pointer. Assume the given Test Data for the computation.

# **Test Data :**

# Input the number of elements to store in the array : 5

# Input 5 number of elements in the array : 5

# element - 1 : 25

# element - 2 : 45

# element - 3 : 89

# element - 4 : 15

# element - 5 : 82

# 7. Consider you have declared and assigned an array variable called Arr1={1,2,2,3,3}; How can you find the unique element present in the given Arr1. Write the program for it.

# 8. Write the program for the below outline

# int Big(int,int,int);

# int main()

# {

# Big(int a,int b,int c);

# }

# int Big(10,20,40)

# {

# /\* write the code for finding the biggest of three value \*/

# }

# 9. Assume the below array variable

# int arr[] = {10, 20, 30, 40, 50, 60};

# How can you find the size of arr[] using pointers? Comment it with suitable code.

# Note:Dont use sizeof().

# 10. Write the program to count the number of digits present in a given number using function recursive method

# 11. Consider the below code,

# int main()

# {

# int x[]={?,?,?,?,?};

# int \*ptr;

# ptr=&x[?];

# printf(“\*(ptr) = %d \n”, \*(ptr));

# printf(“\*(ptr+1) = %d \n”, \*(ptr+1));

# printf(“\*(ptr+2) = %d \n”, \*(ptr+2));

# }

# You assign any 5 values to the array variable x[]. Write the code to determine the value at ptr for any indexing of x initialized by you.

# 12. Write a program in C to count a total number of duplicate elements in an given array. Consider the below Test Data for the computation.

# **Test Data :** Input the number of elements to be stored in the array :3 Input 3 elements in the array : element - 0 : 5 element - 1 : 1 element - 2 : 1

# **Expected Output :**

# Total number of duplicate elements found in the array is : 1

# 13. Consider the below code,

# int main()

# {

# int x[]={?,?,?,?,?};

# int \*ptr;

# ptr=&x[?];

# printf(“\*(ptr) = %d \n”, \*(ptr));

# printf(“\*(ptr+1) = %d \n”, \*(ptr+1));

# printf(“\*(ptr+2) = %d \n”, \*(ptr+2));

# }

# You assign any 5 values to the array variable x[]. Determine the value at ptr for any indexing of x initialized by you.