Assignment question on topic functions in C programming

1. \*\*What is the syntax to declare a function in C?\*\*

Ans : function declaration is also called function prototype

return\_type function\_name(parameter\_type1 param1, parameter\_type2 param2, ...);

ex : int add(int a, int b); // Function prototype (declaration)

1. \*\*Explain the difference between formal parameters and actual parameters.\*\*

Ans :

| **Feature** | **Formal Parameters** | **Actual Parameters** |
| --- | --- | --- |
| **Definition** | Declared in function prototype and definition | Passed during function call |
| **Scope** | Local to the function | Exists in calling function |
| **Memory** | Allocated when function is called | Uses existing memory |
| **Modifiability** | Changes affect only local copies | Original values remain unchanged (unless passed by reference) |

Ex : #include <stdio.h>

// Function with a formal parameter 'num'

void display(int num) {

printf("Number: %d\n", num);

}

int main() {

int value = 10;

display(value); // 'value' is the actual parameter

return 0;

}

1. \*\*How does C handle function return types?\*\*

Ans : In C, **every function must specify a return type**, which determines the type of value the function will return to the caller.

Ex : returning as int

#include <stdio.h>

// Function that returns an integer

int add(int a, int b) {

return a + b; // Returns an integer value

}

int main() {

int result = add(5, 3);

printf("Sum: %d\n", result);

return 0;

}

1. \*\*What is the significance of the `main()` function in C?\*\*

Ans : The main() function is the **entry point** of every C program. When a program is executed, execution starts from main().

Syntax of main

int main() {

// Code execution starts here

return 0;

}

Explanation : int → Return type (main() typically returns an integer).

return 0; → Indicates **successful execution**.

**Why is main() Important?**

✅ **Defines the starting point of execution.**  
✅ **Handles return values to the operating system.**  
✅ **Can accept command-line arguments (argc, argv).**

1. \*\*How do you pass arguments to a function in C?\*\*

Ans : In C, function arguments can be passed in two ways:  
1️. **Pass by Value** (Default) – Function receives a copy of the argument.  
2️ . **Pass by Reference** – Function receives a pointer to the argument.

**Pass by Value (Default)**

* The **function receives a copy** of the actual parameter.
* Changes made inside the function **do not affect** the original value.

**Example: Pass by Value**

#include <stdio.h>

void modify(int x) { // x is a copy of the actual parameter

x = x \* 2; // Changes only local x

}

int main() {

int num = 10;

modify(num); // Pass by value (num remains unchanged)

printf("Original num: %d\n", num); // Output: 10

return 0;

}

🔹 **Even though modify() changes x, num remains unchanged in main().**

**2️. Pass by Reference (Using Pointers)**

* **The function receives a pointer to the actual parameter.**
* **Changes inside the function affect the original variable.**

**Example: Pass by Reference**

**#include <stdio.h>**

**void modify(int \*x) { // x is a pointer to the actual parameter**

**\*x = \*x \* 2; // Modifies the original variable**

**}**

**int main() {**

**int num = 10;**

**modify(&num); // Pass address of num**

**printf("Modified num: %d\n", num); // Output: 20**

**return 0;**

**}**

**🔹 Now, num is modified because modify() operates on its memory address.**

1. **\*\*What is the purpose of the `void` keyword in function parameters?\*\***

**Ans : In C, the void keyword is used in function parameters to indicate:  
1️. A function takes no arguments (void as a parameter).  
2️ . A function does not return a value (void as a return type).**

**void as a Function Parameter (No Arguments)**

* **When void is used in the parameter list, it explicitly states that the function does not accept arguments.**

**#include <stdio.h>**

**// Function with no parameters**

**void greet(void) {**

**printf("Hello, World!\n");**

**}**

**int main() {**

**greet(); // Calling function with no arguments**

**return 0;**

**}**

* **Using void in the parameter list makes it clear that greet() cannot take arguments.**

**void as a Return Type (No Return Value)**

* If void is the return type, the function **does not return a value**.

**void printMessage() {**

**printf("This function returns nothing!\n");**

**}**

**int main() {**

**printMessage(); // Function call**

**return 0;**

**}**

1. **\*\*How can you return multiple values from a function in C?\*\***

**Ans : Pointers (Pass by Reference)  
 Structures (struct)  
 Arrays**

1. **\*\*Explain the concept of function prototypes in C.\*\***

**Ans : A function prototype in C is a declaration of a function before its actual definition. It tells the compiler:  
✅ Function name  
✅ Function return type  
✅ Parameter types (if any)**

**Why Use Function Prototypes?**

* **Allows calling functions before they are defined**
* **Enables type checking (catches mismatched arguments)**
* **Improves code readability**

**Syntax : return\_type function\_name(parameter\_list);**

**Ex : int add(int, int); // Function prototype**

**Return type: int  
 Function name: add  
 Parameters: Two int values**

| **Feature** | **Without Prototype** | **With Prototype** |
| --- | --- | --- |
| **Type Checking** | **❌ No type checking before use** | **✅ Compiler checks parameters** |
| **Order of Definition** | **🚫 Function must be defined before use** | **✅ Can be declared before definition** |
| **Header Files** | **❌ Not suitable for modular programming** | **✅ Encourages modular design** |

1. **\*\*What is the difference between `return 0` and `exit(0)` in the `main()` function?\*\***

**Ans :**

| **Feature** | **return 0;** | **exit(0);** |
| --- | --- | --- |
| **Where It’s Used** | **Inside main()** | **Anywhere in the program** |
| **Function** | **Ends only main() and returns to the OS** | **Terminates the entire program immediately** |
| **Cleanup** | **Calls functions registered with atexit()** | **Calls atexit() functions and cleans up resources** |
| **Exit Code** | **Returns status to OS** | **Terminates with a specified status** |

1. **\*\*Write a C function to find the maximum of two numbers.\*\***

**Ans : #include <stdio.h>**

**// Function to return the maximum of two numbers**

**int max(int a, int b) {**

**return (a > b) ? a : b;**

**}**

**int main() {**

**int num1 = 10, num2 = 20;**

**printf("Maximum: %d\n", max(num1, num2));**

**return 0;**

**}**

**By using if else**

**int max(int a, int b) {**

**if (a > b)**

**return a;**

**else**

**return b;**

**}**

1. **\*\*Implement a C function to calculate the factorial of a number recursively.\*\***

**Ans : #include <stdio.h>**

**// Recursive function to calculate factorial**

**long long factorial(int n) {**

**if (n < 0)**

**return -1; // Error case: Factorial of negative numbers is undefined**

**if (n == 0 || n == 1)**

**return 1; // Base case**

**return n \* factorial(n - 1); // Recursive step**

**}**

**int main() {**

**int num = 5;**

**printf("Factorial of %d is %lld\n", num, factorial(num));**

**return 0;**

**}**

1. **\*\*Write a C program to reverse a string using a function.\*\***

**Ans : Implementation Using Two-Pointer Swap**

**#include <stdio.h>**

**#include <string.h>**

**// Function to reverse a string in-place**

**void reverseString(char \*str) {**

**if (str == NULL) return; // Handle NULL pointer**

**int left = 0, right = strlen(str) - 1;**

**while (left < right) {**

**// Swap characters**

**char temp = str[left];**

**str[left] = str[right];**

**str[right] = temp;**

**left++;**

**right--;**

**}**

**}**

**int main() {**

**char str[] = "Hello, World!"; // Mutable string (Array)**

**printf("Original: %s\n", str);**

**reverseString(str);**

**printf("Reversed: %s\n", str);**

**return 0;**

**}**

**13. \*\*How do you use pointers as function arguments in C?\*\***

**Ans : In C, pointers can be passed to functions to:  
 Modify variables in the calling function (pass by reference)  
 Pass large data structures efficiently  
 Work with dynamic memory**

1. **\*\*Explain the concept of static functions in C.\*\***

**Ans : in C, the static keyword, when used with functions, restricts their scope to the file in which they are defined. This is known as internal linkage, meaning the function cannot be accessed from other files.**

**Uses of Static Functions?**

**Encapsulation – Prevents unintended access from other files  
 Avoids Naming Conflicts – Two files can have functions with the same name  
 Optimizations – The compiler may optimize static functions better since they are limited to one file**

**Static vs. Non-Static Functions**

| **Feature** | **static Function** | **Normal Function** |
| --- | --- | --- |
| **Scope** | **Limited to the file** | **Available across files (extern)** |
| **Encapsulation** | **Yes** | **No** |
| **Naming Conflicts** | **No conflict** | **Can cause conflicts if same name appears in multiple files** |
| **Optimization** | **Compiler may inline** | **Less predictable** |

* **When Should You Use Static Functions?**

**When a function is only needed within a single file**

**To hide implementation details (Encapsulation)**

**When reducing name conflicts in large projects**

1. **\*\*Write a C function to check if a given number is prime.\*\***

**Ans : A prime number is a natural number greater than 1 that is only divisible by 1 and itself.  
Examples: 2, 3, 5, 7, 11, 13, 17, ...**

**#include <stdio.h>**

**#include <math.h>**

**// Function to check if a number is prime**

**int isPrime(int num) {**

**if (num < 2) return 0; // Numbers < 2 are not prime**

**if (num == 2) return 1; // 2 is the only even prime number**

**if (num % 2 == 0) return 0; // Exclude all even numbers**

**for (int i = 3; i <= sqrt(num); i += 2) {**

**if (num % i == 0) return 0; // Found a divisor, not prime**

**}**

**return 1; // Prime number**

**}**

**int main() {**

**int num;**

**printf("Enter a number: ");**

**scanf("%d", &num);**

**if (isPrime(num))**

**printf("%d is a prime number.\n", num);**

**else**

**printf("%d is not a prime number.\n", num);**

**return 0;**

**}**

**Programming questions using functions in C:**

**1. \*\*Write a C program to find the sum of digits of a number using a function.\*\***

**Ans : #include <stdio.h>**

**void main()**

**{**

**int num, r, sum=0;**

**printf("Enter a number: ");**

**scanf("%d", &num);**

**while (num>0)**

**{**

**r=num%10;**

**sum=sum+r;**

**num=num/10;**

**}**

**printf("Sum of digits of %d\n", sum);**

**}**

**Extract last digit → num % 10**

1. **Add to sum**
2. **Remove last digit → num /= 10**
3. **Repeat until num becomes 0**

**2. \*\*Implement a function to find the length of a string in C.\*\***

**Ans : #include <stdio.h>**

**#include <string.h>**

**int main() {**

**char str[100];**

**printf("Enter a string: ");**

**scanf("%s", str);**

**printf("Length of string \"%s\" = %lu\n", str, strlen(str));**

**return 0;**

**}**

1. **\*\*Write a C program to check if a given string is a palindrome using functions.\*\***

**Ans :** **A palindrome is a string that reads the same forward and backward.  
Examples: "madam", "racecar", "level", "radar"**

**#include <stdio.h>**

**#include <string.h>**

**// Function to check if a string is a palindrome**

**int isPalindrome(char str[]) {**

**int left = 0, right = strlen(str) - 1;**

**while (left < right) {**

**if (str[left] != str[right])**

**return 0; // Not a palindrome**

**left++;**

**right--;**

**}**

**return 1; // Palindrome**

**}**

**int main() {**

**char str[100];**

**// Input string from user**

**printf("Enter a string: ");**

**scanf("%s", str); // Reads a single word (use fgets for multi-word input)**

**// Function call and output**

**if (isPalindrome(str))**

**printf("\"%s\" is a palindrome.\n", str);**

**else**

**printf("\"%s\" is not a palindrome.\n", str);**

**return 0;**

**5.\*\*Write a program to calculate the power of a number using a recursive function in c?**

**Ans : #include <stdio.h>**

**int power (int, int);//function declaration**

**int main()**

**{**

**int b,p,r;//base power result**

**printf("enter base value");**

**scanf("%d", &b);**

**printf("enter power value");**

**scanf("%d", &p);**

**r=power(b,p);//function call**

**printf("result=%d",r);**

**return 0;**

**}**

**int power (int b, int p)//function definition**

**{**

**if (p==0)//base condition it is checking for the recursion function**

**return 1;**

**else**

**return b\*power(b,p-1);// recursive function call**

**}**

**6. \*\*Implement a function to sort an array of integers in ascending order using the bubble sort algorithm in C.\*\***

**Ans : #include <stdio.h>**

**// Function to perform Bubble Sort**

**void bubbleSort(int arr[], int n) {**

**int i, j, temp;**

**int swapped;**

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

**swapped = 0; // Track if any swaps occur in this pass**

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

**if (arr[j] > arr[j + 1]) { // Swap if out of order**

**temp = arr[j];**

**arr[j] = arr[j + 1];**

**arr[j + 1] = temp;**

**swapped = 1; // Mark that a swap happened**

**}**

**}**

**// If no swaps, array is already sorted**

**if (swapped == 0)**

**break;**

**}**

**}**

**// Function to print the array**

**void printArray(int arr[], int n) {**

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

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

**}**

**printf("\n");**

**}**

**int main() {**

**int arr[] = {64, 34, 25, 12, 22, 11, 90};**

**int n = sizeof(arr) / sizeof(arr[0]);**

**printf("Original array: ");**

**printArray(arr, n);**

**bubbleSort(arr, n);**

**printf("Sorted array: ");**

**printArray(arr, n);**

**return 0;**

**}**

**7. \*\*Write a C program to find the GCD (Greatest Common Divisor) of two numbers using a function.\*\***

**Ans : #include <stdio.h>**

**int gcd(int a, int b) //recursive function used to find GCD**

**{**

**if (b == 0)**

**return a; // Base case: when remainder is 0**

**return gcd(b, a % b); // Recursive step**

**}**

**int main()**

**{**

**int num1, num2;**

**printf("Enter two numbers: ");//input from user**

**scanf("%d %d", &num1, &num2);**

**printf("GCD of %d and %d is %d\n", num1, num2, gcd(num1, num2));//function call and output**

**return 0;**

**}**

**8. \*\*Create a function to calculate the area of a circle given its radius in C.\*\***

**Ans :** **#include<stdio.h>**

**void main()**

**{**

**int r;**

**float pi=3.14,area;**

**printf("enter the radius of circle:");**

**scanf("%d",&r);**

**area=pi\*r\*r;**

**printf("area of circle=%f",area);**

**}**

**9. \*\*Write a program to check if a given year is a leap year or not using a function in C.\*\***

**Ans : A leap year occurs:  
✔ If the year is divisible by 4 AND not divisible by 100  
✔ OR if the year is divisible by 400**

**#include <stdio.h>**

**void main() {**

**int year;**

**printf("enter the year:");**

**scanf("%d", &year);**

**if(year %400 == 0)**

**printf("%d is a leap year ", year);**

**else if(year%100 ==0)**

**printf("%d is not a leap year ", year);**

**else if (year %4 ==0)**

**printf("%d is a leap year ", year);**

**else**

**printf("%d is not a leap year ", year); }**

**10. \*\*Implement a function to reverse an array of integers in C.\*\***

**Ans : #include <stdio.h>**

**void main()**

**{ int a[5] , i;**

**printf("enter array elements:");**

**for (i=0;i<=4;i++)**

**{**

**scanf ("%d", &a[i]);**

**}**

**printf(" reverse array elements:");**

**for(i=4;i>=0;i--)**

**{**

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

**}**

**}**

**11. \*\*Write a C program to convert temperature from Celsius to Fahrenheit using a function.\*\***

**Ans : #include <stdio.h>**

**void main()**

**{**

**float c ,fh;**

**printf("enter the temperature in centigrade:");**

**scanf("%f", &c);**

**fh=(c\*9/5)+32;**

**printf("temperature in fahrenheit is %f\n", fh);**

**}**

**12. \*\*Create a function to calculate the sum of elements in a 2D array in C.\*\***

**Ans : #include<stdio.h>**

**void main()**

**{**

**int a[3][3],i,j,sum=0;**

**for(i=0;i<3;i++)**

**for(j=0;j<3;j++)**

**{**

**printf ("\nenter a number:");**

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

**}**

**for(i=0;i<3;i++)**

**{**

**printf("\n");**

**for(j=0;j<3;j++)**

**printf("%d\t", a[i][j]);**

**}**

**for(i=0;i<3;i++)**

**for(j=0;j<3;j++)**

**sum=sum+a[i][j] ;**

**printf("total sum is %d", sum);**

**}**

**13. \*\*Implement a function to concatenate two strings in C.\*\***

**Ans : String concatenation involves appending one string to another**

**#include <stdio.h>**

**void concatenateStrings(char str1[], char str2[])**

**{**

**int i = 0, j = 0;**

**// Find the end of str1**

**while (str1[i] != '\0') {**

**i++;**

**}**

**// Append str2 to str1**

**while (str2[j] != '\0') {**

**str1[i] = str2[j];**

**i++;**

**j++;**

**}**

**str1[i] = '\0';// null Null-terminate the concatenated string**

**}**

**int main() {**

**char str1[100], str2[50];**

**printf("Enter first string: ");**

**scanf("%s", str1);**

**printf("Enter second string: ");**

**scanf("%s", str2);**

**concatenateStrings(str1, str2);// function call**

**printf("Concatenated string: %s\n", str1);//output result print here**

**return 0;**

**}**

**14.\*\*Write a program to find the factorial of a number using recursion and iteration, both implemented as separate functions in C.\***

**Ans : factorial of a number using recursion**

**//factorial of a number using recursion**

**#include <stdio.h>**

**int factorial(int x);**

**int main()**

**{**

**int num,fact;**

**printf("enter the number:");**

**scanf("%d",&num);**

**fact=factorial(num);**

**printf("factorial of %d is %d", num,fact);**

**return 0;**

**}**

**int factorial(int x)**

**{**

**if(x==1)**

**return 1;**

**return x\*factorial(x-1);**

**}**

**Factorial of number using iteration function**

**//factorial of a number using iteration**

**#include <stdio.h>**

**long long factorialIterative(int n) {**

**long long fact = 1;**

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

**fact \*= i;**

**}**

**return fact;**

**}**

**int main() {**

**int num;**

**printf("Enter a number: ");//input from user**

**scanf("%d", &num);//read the address of given number**

**if (num < 0) {**

**printf("Factorial is not defined for negative numbers.\n");**

**} else {**

**printf("Factorial of %d is: %lld\n", num, factorialIterative(num));//factorial function will done by using iterstion**

**}**

**return 0;**

**}**

1. **\*\*Create a function to find the median of three numbers in C.\*\***

**Ans : #include <stdio.h>**

**// Function to find the median of three numbers**

**int findMedian(int a, int b, int c) {**

**if ((a > b && a < c) || (a > c && a < b))**

**return a;**

**else if ((b > a && b < c) || (b > c && b < a))**

**return b;**

**else**

**return c;**

**}**

**int main() {**

**int num1, num2, num3;**

**// Input three numbers**

**printf("Enter three numbers: ");**

**scanf("%d %d %d", &num1, &num2, &num3);**

**// Function call**

**int median = findMedian(num1, num2, num3);**

**// Output result**

**printf("The median is: %d\n", median);**

**return 0;**

**}**

1. **\*\*Write a C program to generate the Fibonacci series up to a given number using a function.\*\***

**Ans :** **#include <stdio.h>**

**int main()**

**{**

**//0 1 1 2 3 5 8......n**

**int a=0,b=1,c,n,i;**

**printf("enter the number of terms:");**

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

**for(i=1;i<=n;i++)//function call**

**{**

**printf("%d ", a);**

**c=a+b;//0+1=1 this 1 will be store in c**

**a=b;//variable b value store in a**

**b=c;//variable c value store in b**

**}**

**return 0;**

**}**

1. **\*\*Implement a function to check if a given string is an anagram of another string in C.\*\***

**Ans : Two strings are anagrams if they contain the same characters in the same frequency, but order of character is differ.is called anagram**

* **Ex :” listen" and "silent" → ✅ Anagrams**
* **"triangle" and "integral" → ✅ Anagrams**
* **"apple" and "pale" → ❌ Not Anagrams**

** Check Lengths:**

* **If lengths differ, the strings cannot be anagrams.**

** Character Frequency Count:**

* **Use an array of size 256 (for ASCII characters).**
* **Increment for str1, Decrement for str2.**
* **If all values in the count array are 0, the strings are anagrams.**

**// check if a given string is an anagram of another string in C.\*\***

**#include <stdio.h>**

**#include <string.h>**

**void main ()**

**{**

**char str1[1000],str2[1000];**

**int len1,len2,len,i,j,found=0,notfound=0;**

**printf("Enter first string: ");**

**scanf("%s", str1);**

**printf("Enter second string: ");**

**scanf("%s", str2);**

**len1=strlen(str1);**

**len2=strlen(str2);**

**if(len1==len2)**

**{**

**len=len1;**

**for(i=0;i<len;i++)**

**{**

**for(j=0;j<len;j++)**

**{ if(str1[i]==str2[j])**

**{**

**found=1;**

**break;**

**}**

**}**

**if (found==0)**

**{**

**notfound=1;**

**break;**

**}**

**}**

**if (notfound==1)**

**{ printf("not anagram");}**

**else**

**{ printf("an anagram");}**

**}**

**else**

**{printf("string size are different");}**

**}**

1. **\*\*Write a program to print all prime numbers within a given range using a function in C**

**Ans: A prime number is a number greater than 1 that has only two divisors: 1 and itself.  
Examples: 2, 3, 5, 7, 11, 13, ...**

**#include <stdio.h>**

**int isprime(int n)**

**{ int i=2;**

**for(i=2;i\*i<n;i++)**

**{**

**if(n%i==0)**

**{**

**return 0;**

**}**

**}**

**return 1;**

**}**

**int main() {**

**int n1,n2,i;**

**printf("enter first number\n");**

**scanf("%d",&n1);**

**printf("enter second number\n");**

**scanf("%d",&n2);**

**printf("the prime numbers between %d and %d are:", n1,n2);**

**for(i=n1;i<=n2;i++)**

**{**

**if(isprime(i))**

**{ printf("%d\t", i);**

**}**

**} return 0; }**

**Output : the prime numbers between 2 to 27 are 2 3 4 5 7 9 11 13 17 19 23 25**

1. **\*\*Create a function to calculate the area of a triangle given its base and height in C.\*\***

**Ans : Formula for the Area of a Triangle:**

**Area=1/2 \* base \* height**

**#include<stdio.h>**

**void main()**

**{**

**float base,height,area;**

**printf("enter base and height of a triangle");**

**scanf("%f%f",&base,&height);**

**area=0.5\*base\*height;**

**printf("area of a triangle if %f", area);**

**}**

1. **\*\*Write a C program to find the factorial of a large number using an array and a function.\*\***

**Ans : #include<stdio.h>**

**void factorial(int);//function declaration**

**int main()**

**{**

**int num;**

**printf("enter positive number to find factorial\n");**

**scanf("%d",&num);**

**factorial (num);**

**return 0;**

**}**

**void factorial(int num);**

**{**

**int count fact=1;**

**if(num==0)**

**{ printf("factorial of 0 is 1");**

**}**

**else**

**{**

**for (count=0;count<=num;count++)**

**{**

**fact=fact\*count;**

**}**

**printf("factorial of %d is %d", num, fact);**

**}**

**}**