USN | | | | | | | | | | |

Internal Assessment Test 3 – MAR 2022

Sub:	Problem solving through Programming					Sub Code:	21PSP13	Branch:	I Year
Date:	30.03.2022	Duration:	90 mins	Max Marks:	50	Sem/Sec:	I Sem (I, J, K, L, M, N,O)		

	Answer any FIVE FULL Questions	MARKS

1 (a)	Discuss the various ways of passing parameter to the functions
	❖ There are mainly two parameter passing mechanisms in 'C' functions. They are:
	i. Call by Value or Pass by Value
	ii. Call by Address or Pass by Address
	Call by Value or Pass by Value
	❖ Calling (Invoking) a function by passing values or variables to the function is called as call by value.
	❖ The values of actual parameters are copied into formal parameters
	❖ Formal parameters contain only the copy of actual parameters. So, even if the values of the formal parameters change in the called function, the values of the actual parameters are not changed.
	Ex: #include<stdio.h>
	int add (int a, int b) // Formal parameters a = 10, b =20
	{
	int sum;
	sum = a + b;
	a=3;
	b=5;
	printf(" Called function value of a and b is %d%d\t",a,b);
	return sum;
	}
	void main(){
	int m=10,n=20, res;
	printf("before passing value of m and n is %d%d\t",m,n);
	res =add(m,n); // Actual parameter m=10, n =20
	printf("result = %d \n", res);
	printf("after passing value of m and n is %d%d\t ",m,n);
	}
	
	ii. Call by Address or Pass by Address
	❖ Call by address is done indirectly by passing the address of the variables in the function call and changing the value of the variable through its address.
	❖ In the called function, the formal parameters should be declared as pointers with the same type as the actual parameters.
	❖ The addresses of actual parameters are copied into formal parameters. Using these addresses the values of the actual parameters can be changed.
	❖ In call by reference if any change in formal parameters imply then there is a change in actual parameters.
		[5]

	Ex: #include<stdio.h>
	int add (int *a, int *b) // Formal parameters *a = 10, *b =20
	{
	int sum;
	sum = *a + *b;
	*a=3;
	*b=5;
	printf(" Called function value of a and b is %d%d\t",*a,*b);
	return sum;
	}
	void main()
	{
	int m=10,n=20, res;
	printf("before passing value of m and n is %d%d\t",m,n);
	res = add(&m, &n); // Actual parameter m=10, n =20
	printf("result = %d \n", res);
	printf("after passing value of m and n is %d%d\t",m,n);
	}	

1 (b)	How do you explain the declaration and initialization of string variable ❖ A string is a sequence of characters enclosed within double quotes". **Declaring String variables** ❖ A string is declared like an array of characters. **Syntax: char string_name[size/length];** **Example: char name[21];** ● Size of the string is 21, means that it can store up to 20 characters plus one null character. **Initializing the Strings** We can initialize an array of characters (Strings) in the declaration itself. **Examples**: 1. **char a[9]={'C', 'O', 'M', 'P', 'U', 'T', 'E', 'R', '\0'};** The compiler allocates 9 memory locations ranging from 0 to 8 and these locations are initialized with the characters in the order specified. 2. **char b[ ]={'C', 'O', 'M', 'P', 'U', 'T', 'E', 'R', '\0'};** For this declaration, the compiler will set the array size to the total number of initial values. i.e. 9. The characters will be stored in these memory locations in the order specified 3. **char b[ ]= "COMPUTER";** Here, the string length is 8 bytes. But string size is 9 bytes. So, the compiler reserves 8+1 memory locations and these locations are initialized with the characters in the order specified. The string is terminated by '\0' by the compiler.	[5]
2(a)	Define a function. List various advantages of a function. ❖ **"The set of instructions that performs some specific, well-defined task is called a Function."** **or** ❖ **"Function is a small program or program segment that carries out some specific well-defined tasks".** ❖ **Advantages of Functions**	[5]

	▪ i. Reduces the Complexity. ▪ When a program contains more instructions (complex program), then such a large program can be divided into a number of subprograms called functions. ▪ ii. Improves Readability. ▪ iii. Easy to debug the errors. ▪ iv. Reusability: The set of instructions specifying the task performed by the function is written just once but it can be used any number of times. ▪ v. Easy to do the modifications.	

2(b)	Write a C program to find GCD and LCM of two numbers using concept of functions
	/* GCD using recursion */
	#include <stdio.h>
	int gcd(int a,int b) {
	int rem=a%b;
	if(rem==0)
	return (b);
	else
	return (gcd(b,rem));
	}
	int main() {
	int x,y,hcf;
	printf("Enter two numbers : ");
	scanf("%d%d",&x,&y);
	hcf=gcd(x,y);
	printf("GCD = %d\n",hcf);
	printf("LCM = %d\n",x*y/hcf);
	return (0);
	}
	Output: Enter two numbers : 45 15
	GCD = 15
	LCM = 45
		[5]

3(a)	What is recursion? Write a C program to computer factorial using recursion
	❖ "The process in which a function calls itself again and again is called Recursion".
	❖ A function which calls itself again and again is called a Recursive function.
	❖ While using recursion, users need to be careful to define exit conditions from function; otherwise it will go in an infinite loop.
	❖ Recursive functions are very useful to solve many mathematical problems like calculating factorials of a number, generating Fibonacci series, etc.
	#include<stdio.h>
	int factorial(int n)
	{
	if(n==1)
	return 1;
	else
	return (n*fact(n-1));
		[5]

```
}
 void main()
 {
        int n,fact;
        printf("Enter a number=");
        scanf("%d",&n);
        fact=factorial(n);
        printf("\nFactorial of given number=%d",fact);
 }
```

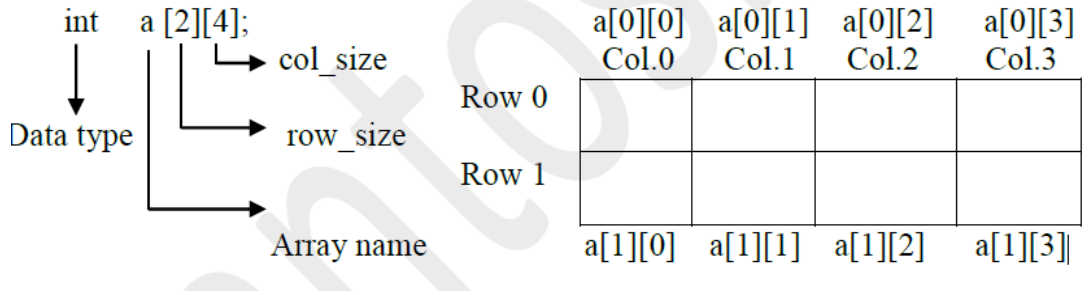
Output: Enter number: 5

Factorial of 5 = 120

3(b) How is a 2D array represented in memory? Explain with a suitable example [5]

- Arrays which are specified with 2 subscripts (2 set of square brackets [ ][ ]) are called 2-dimensional arrays.
- Arrays with two or more dimensions are called Multi-dimensional arrays.
- In 2-Dimensional Array, the first index indicates the 'row size'(the number of rows) and second index indicates the 'column size'(the number of columns).
  - Ex: int a[10][10]; //Two-dimensional array
- As we declare the variables before they are used in a program, an array must also be declared before it is used using the following syntax.
- Syntax:
  - **data_type array_name [row_size][col_size];**

As we initialize a variable to the required value, we can initialize the individual elements of the array during initialization.



Syntax:

data_type array_name[row_size][col_size]={ {a1,a2,-----,an}, {b1,b2,-----,bn},..., {z1,z2,-----,zn} };

```
int a[4][3]= {{11,22,33}, {44,55,66},{77,88,99},{10,20,30} };
```

4 Demonstrate the use of C user-defined functions with a suitable example [10]

- The functions written by the programmer/user to do the specific tasks is called User-Defined/ Programmer Defined Functions.
- Elements of User-Defined Functions
- The three elements of user-defined functions are shown below:
  - i.Function Prototype/Declaration
  - ii. Function Definition
  - iii. Function Call

i. Function Prototype/Declaration

- As we normally declare the variables before they are used, the functions also should be declared before they are used.
- The process of declaring the functions before they are used (or called) in the

	program is called Function Prototype.
	❖ The function prototype is also called as Function declaration.
	❖ It does not contain the body of the function.
	■ Syntax: **return_type function_name(parameter list);**
	❖ Ex: int add(int a, int b);
	❖ The function prototype contains the following information in a single line terminated by semicolon:
	■ The type of the value returned by the function.
	■ The name of the function.
	■ The number of parameters passed to that function.
	■ The type of each parameter.
	ii. Function Definition
	❖ The program module that is written to achieve a specific task is called function definition.
	❖ Each function definition consists of two parts:
	■ Function Header
	■ Function Body
	● Syntax: return_type function_name (parameter list)
	{
	declaration part;
	executable part;
	return statement;
	}
	❖ **return statement:** It is a keyword used to return the control to the calling function (main) with/without a value.
	i. Function Call
	❖ Once the function is defined, it has to be called so as to achieve the task. This method of calling a function to achieve a specified task is called Function Call.
	❖ The function can be called by writing the name of the function and passing the appropriate number of arguments.
	❖ The number of arguments in the function call and number of parameters in the function definition must match.
	❖ Also the order of arguments in the function call and parameters in the function definition must match.
	■ Example: add(m,n);	

5	Using suitable code, discuss the working of the following string functions: i.strcat- Syntax: **strcat(s1,s2);** ❖ The 'strcat()' function is used to concatenate or join the two strings. ❖ The 'strcat()' function copies all the characters of string s2 to the end of string s1. The NULL character of string s1 is replaced by the first character of s2. ii.strcmp-Syntax **strcmp(s1,s2);** ❖ This function is used to compare two strings. ❖ The comparison starts with first character of each string. The comparison continues till the corresponding characters differ or until the end of the character is reached. ❖ The following values are returned after comparison: 1. If two strings are equal, the function returns 0. 2. If s1 is greater than s2, a positive value is returned.	[10]

3. If s1 is less than s2, then the function returns a negative value.

iii.strcpy- Syntax: **strcpy(dest,src);**

- The 'strcpy()' function copies the contents of source string src to destination string dest including '\0'.

The strcpy() function copies characters from the source string to the destination string until it finds null character.

iv.strlen-The 'strlen()' function can be used to find the length of the string in bytes.

- This function calculates the length of the string up to but not including the 'null character'.
  - Syntax:**length=strlen(str);**

Where,

- 'str' is a string.
- 'length' is an integer representing the length of 'str' in bytes excluding the null character.

- The 'sizeof' operator can be used to determine the size of a declared string.
  - Syntax: **sizeof(str);**

v.strncmp-Syntax: **strncmp(s1,s2,n);**

This function is used to compare the first n number of characters in two strings.
The comparison starts with the first character of each string. The comparison continues till the corresponding characters differ or until the end of the character is reached or specified numbers of characters have been tested.

```
#include<stdio.h>
#include<string.h>
void main()
{
  //Strlen
char str[10]= "RAMA";
int length;
length=strlen(str);
printf("Length of the string is=%d\n",length);
   //'sizeof' operator can be used to determine the size
of a declared string.
printf("Size of string is=%d\n",sizeof(str));

  // strcpy
char src[10]= "computer",dest[10];
strcpy(dest,src);
printf("The    Source    String=%s\n    The    Destination
String=%s\n",src,dest);

  //strncpy
  char src1[10]= "Computer", dest1[10];
strncpy(dest1,src1,3);
printf  ("The Source String=%s\n The Destination using n
value String=%s \n",src1,dest1);

  //strcat
  char s1[15]= "Good";
char s2[15]= "Morning";
```

```
strcat(s1,s2);
printf("The concatenated String=%s\n",s1);

  //strncat(n)
  char str3[15]= "Good";
char str4[15]= "Morning";
strncat(str3,str4,4);
printf("The concatenated String  using n =%s\n",str3);

  //strcmp
  char s5[10]="Hello";
char s6[10]="Hello";
if(strcmp(s5,s6)==0)
printf("The two strings are identical\n");
else
printf("The two strings are not identical\n");
}
```

6(a) How is the structure in C different from union? Give example [5]

Structure	Union
1. It can be defined using **struct** keyword.	It can be defined using a **union** keyword.
2. Every member within structure is assigned a unique memory location.	In union, a memory location is shared by all the data members.
3. Changing the value of one data member will not affect other data members in structure.	Changing the value of one data member will change the value of other data members in union.
4. It allows initializing several members at once.	It allows initializing only the first member of union.
5. The total size of the structure is the sum of the size of every data member.	The total size of the union is the size of the largest data member.
6. It is used for storing various data types.	It is used for storing one of the many data types that are available.
7. It reserves space for each and every member separately.	It reserves space for a member having the highest size.
8. Any member can be retrieve at a time.	Only one member can be retrieve at a time.

Example: struct student

```
{
int rollno;
char name[50];
string phone;
};
union Student {

char name[32];
int age;
string email;
};
```

6(b) What is a pointer? Write a C program to swap two integer values using pointers [5]

	❖ A pointer is a derived data type in C. it is built from one of the fundamental data types available in C.
	A pointer is a variable that holds address of other variable. Since these memory addresses are the locations in the computer memory where program instructions and data are stored, pointers can be used to access and manipulate data stored in the memory.
	#include<stdio.h>
	void swap(int *a, int *b);
	void main()
	{ int x=10,
	y=20;
	printf("before swapping\n x=%d \n y=%d\n\n", x, y);
	swap(&x,&y); //function call, add of 2199823
	printf("after swapping: \n x = %d \n y = %d",x,y);
	}
	void swap(int *a,int *b)
	{ int temp;
	temp=*a; //temp=10
	*a=*b; //a=20
	*b=temp; //b=10
	}	

7(a)	Define a structure by name DoB consisting of three variable members dd, mm and yy of type integer. Develop a C program that would read values to the individual member and display the date in mm/dd/yy form.
	#include <stdio.h>
	struct dob
	{
	int dd,mm,yy;
	};
	struct dob s;
	void main()
	{
	int i;
	
	printf("Enter the DOB details");
	printf("Enter the date\n");
	scanf("%d",&s.dd);
	printf("Enter the month\n");
	scanf("%d",&s.mm);
	
	printf("Enter the year\n");
		[5]

	scanf("%d",s.yy);
	
	printf("DOB Details are:\n");
	printf("%02d/%02d/%04dn", s.mm, s.dd, s.yy);
	
	}	

7(b)	Discuss the use of #define and #include in C programming **#include** ❖ #include loads specified files before compilation of user written program. ■ Syntax: #include<header_file_name.h> or #include "header_file_name.h" ■ Example: #include<stdio.h> or #include "stdio.h" ■ stdio.h has macros like stdin, stdout, stderr etc. ctype.h has macros like isalpha(x), islower(x) etc **#define** ❖ #define defines a macro substitution. ❖ It allows you to replace an identifier (variable, function names) in a program by a predefined string. ■ Syntax: #define identifier string ■ Example: #define pi 3.142 //where pi is a variable and 3.142 is variable value	[5]