Solution with scheme-Model Answer

Prof.Lynsha Helena Pratheeba HP/Prof.Rajeshwari R/ Prof.Kavyashri/Prof.Reshma										
Internal Assessment Test 2 – January 2025										
Sub	**Principles of Programming Using C**					Sub code	**BPOPS103**	Branch	**ISE, AIML, CSE(AIML), AIDS**	
Date	**21.01.2025**	Duration	**90 mins**	Max Marks	**50**	Sem /Sec	**I Sem P-Cycle (A- H)**		**OBE**	

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1.	What is Recursive function? Write C program to display Fibonacci series using recursion.	[10]	CO3	L3
2.	Write a C program to implement Matrix multiplication and validate the rules of multiplication.	[10]	CO3	L3
3. a)	Explain declaration and initialization of 1-dimensional arrays.	[5]	CO3	L2
b)	Explain declaration and initialization of 2-dimensional arrays.	[5]	CO3	L2
4.	Write a C program to implement binary search on unsorted array.	[10]	CO3	L4
5.	Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers.	[10]	CO4	L3
6. a)	Define pointers.Explain passing parameters to functions using pointers.	[5]	CO4	L2
b)	Differentiate Null pointer and Generic pointer.	[5]	CO4	L2
7.	Implement structures to read, write and compute average- marks of the students, list the students scoring above and below the average marks for a class of N students.	[10]	CO4	L3
8.	Define string. List out any 5 string manipulation built-in functions with examples.	[10]	CO4	L2

1.	What is Recursive function? Write C program to display Fibonacci series using recursion. **Definition [1] +Program explanation[1]+Program [8]**	[10]

Definition :
A recursive function is a function that calls itself to solve a smaller version of its task until a final call is made which does not require a call to itself.

Program explanation:

Fibonacci Series in C: 0, 1, 1, 2, 3, 5, 8, 13, 21 etc
The Fibonacci series is a sequence of numbers in which each number is the sum of the two preceding ones. The sequence typically starts with 0 and 1, and the subsequent numbers are generated by adding the two previous numbers. The first few numbers in the Fibonacci sequence are:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...
In general, the Fibonacci sequence is defined by the recurrence relation:

- F(0) = 0
- F(1) = 1
- F(n) = F(n-1) + F(n-2) for n > 1

Program

```
#include<stdio.h>
int fib(int)
int main()
{
int n, i = 0, res;
printf(“Enter the number of terms\n”)
scanf("%d",&n);
printf("Fibonacci series\n")
for ( i= 0 ; i< n ; i++ )
{
res=fib(int i)
printf(“%d\n”,res)
}
return 0;
}
Int fib(int n
{
if ( n == 0 )
return 0;
else if ( n == 1 )
return 1;
else
return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

Output
Enter the number of terms :5
Fibonacci series:0, 1, 1, 2, 3, 5

Write a C program to implement Matrix multiplication and validate the rules of multiplication.
Program explanation[9]+output [1]

2.

```
#include<stdio.h>
int main()
{
int a[10][10],b[10][10],c[10][10];
int m,n,p,q;
int i,j,k;
// Input the order of Matrix A - m x n
printf("Enter the order of matrix A :");
scanf("%d%d",&m,&n);
// Input the order of Matrix B - p x q
printf("Enter the order of matrix B:");
scanf("%d%d",&p,&q);
if(n!=p)
{
printf("Number of columns of Matrix A is not equal to number of rows of matrix B\n");
printf("Matrix Multiplication not possible....\n");
return (1);
}

// Input the elements into Matrix A
printf("\nEnter %d elements into matrix A : ", m*n);
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&a[i][j]);
}
}

// Display matrix A in matrix format
printf("\nThe  matrix A is ---\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",a[i][j]);
}
printf("\n");
}

// Input the elements into Matrix B
printf("\nEnter %d elements into matrix B : ", p*q);
for (i=0;i<p;i++)
{
for (j=0;j<q;j++)
{
scanf("%d",&b[i][j]);
}
}

// Display Matrix B in matrix format
printf("\nThe matrix B is ---\n");
for(i=0;i<p;i++)
{
for(j=0;j<q;j++)
{
```

[10]

```
printf("%d\t",b[i][j]);
}
printf("\n");
}

// Compute (Matrix A) X (Matrix B)
for(i=0;i<m;i++)
{
for(j=0;j<q;j++)
{
c[i][j] = 0;
for(k=0;k<n;k++)
{
c[i][j] = c[i][j] + (a[i][k] * b[k][j]);
}
}
}

// Display product matrix - Matrix C
printf("\nThe product matrix is ---\n\n");
for(i=0;i<m;i++)
{
for(j=0;j<q;j++)
{
printf("%d\t",c[i][j]);
}
printf("\n");
}
return 0;
}
```

OUTPUT:.

```
$cc prog5.c
$./a.out
Enter the order of matrix A :2 2
Enter the order of matrix B:2 2
Enter 4 elements into matrix A : 1 2 3 4
The matrix A is ---
1 2
3 4
Enter 4 elements into matrix B : 1 3 2 4
The matrix B is ---
1 3
2 4
The product matrix is ---
5 11
11 25
```

Explain declaration and initialization of 1-dimensional arrays.

Syntax[1]+declaration[2]+initialization [2]

3.)	Syntax for declaring a 1-dimensional array:
	data_type array_name[size];
	
	Examples:
	
	int numbers[5]; // Declares an array of 5 integers;
	
	Initialization of one-dimensional array may be done directly or indirectly running a loop.
	Examples of direct initialization:
	
	int numbers[5] = {1, 2, 3, 4, 5}; // Initializes the array with 5 values
	
	Example of indirect initialization:
	#include<stdio.h>
	void main()
	{
	int i,a[5];
	printf("Enter the elements: ")
	for(i=0;i<5;i++)
	{
	scanf("%d",&a[i]);
	}
	for(i=0;i<5;i++)
	{
	printf("Array a[%d]=%d\n",i,a[i])
	}
	
	Output:
	Enter the elements: 1 2 3 4 5
	Array a[0]=1
	Array a[1]=2
	Array a[2]=3
	Array a[3]=4
	Array a[4]=5
		[5]

3.b)	Explain declaration and initialization of 2-dimensional arrays.
	Syntax[1]+declaration[2]+initialization [2]
	
	A two dimensional array is specified using two subscripts where one subscript denotes row and the other denotes column.
	
	• C treats a two dimensional array as an array of one dimensional arrays.
	• A two dimensional array is declared as:
	
	Syntax for declaring a 1-dimensional array:
	data_type array_name[row_size][column_size];
	Example: int marks[3][5];
	
	Initialization of 2-D arr
	1. Initialize with total number of elements
	int a[2][3]={1,2,3,4,5,6}
	2. Initialize with set
	int a[2][3]={{1,2,3},{4,5,6}
	
	Write a program to print the elements of a 2D array
	#include<stdio.h>
	#include<conio.h>
	main()
	{
	
	int arr[2][2] = {12, 34, 56,32};
	int i, j;
	for(i=0;i<2;i++)
	{
	
	printf("\n");
	for(j=0;j<2;j++)
	
	printf("%d\t", arr[i][j]);
	
	}
	return 0;
	
	}
	
	Output
	12 34
	56 32
		[5]

4. Write a C program to implement binary search on unsorted array. [10]

Program [9]+output [1]

```
#include <stdio.h>
int main()
{
 // Define variables
 int a[20];
 int n,i,j,temp,key;
 int first,mid,last;

 // Accept the size of the array
 printf("Enter the size of the array :");
 scanf("%d",&n);

 // Accept elements into the array
 printf("Enter %d elements :",n);
 for(i=0;i<n;i++)
 {
 scanf("%d",&a[i]);
 }

 // Print the elements of the array before sorting
 printf("The elements of the array before sorting is ----\n");
 for(i=0;i<n;i++)
 {
 printf("%d\t",a[i]);
 }

 /* Sort the array as the data
   must be sorted for binary search
 */

 for(i=0;i<n-1;i++)
 {
 for(j=0;j<n-1-i;j++)
 {
 if(a[j]>a[j+1])
 {
 temp=a[j];
 a[j]=a[j+1];
 a[j+1]=temp;
 }
 }
 }

 // Display the sorted elements of the array
 printf("\n\nThe sorted array is ---\n");
 for(i=0;i<n;i++)
 {
 printf("%d\t",a[i]);
 }

 // Accept the element to be searched
 printf("\n\nEnter the element to be searched :");
 scanf("%d",&key);
```

```
 // search for the element in the sorted array
 first=0;
 last=n-1;

 while(first <= last)
 {
 mid=(first+last)/2;
 if(key==a[mid])
 {
 printf("\nThe element %d is found at location %d\n",key,mid+1);
 return (0);
 }
 else if (key < a[mid])
 {
 last = mid-1;
 }
 else
 {
 first = mid+1;

 }
 }

 printf("\nThe element %d is not found in the array\n",key);
 return (1);
 }
 $ ./a.out
Enter the size of the array :5
Enter 5 elements :87
0
4
2
3
The elements of the array before sorting is ----
87 0 4 2 3
The sorted array is ---
0 2 3 4 87
Enter the element to be searched :0
The element 0 is found at location 1
```

5.	Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers.
	Program [9]+output [1]
	
	#include <stdio.h>
	#include <math.h>
	
	int main()
	{
	int i, n;
	float a[10],sum,mean,var,sd;
	float *p;
	printf("Enter Number of elements:");
	scanf("%d", &n);
	printf("Enter %d numbers :",n);
	p = a;
	for (i=0;i<n;i++)
	{
	scanf("%f", p);
	p++;
	}
	sum = mean = var = sd = 0.0;
	p = a;
	for(i=0;i<n;i++)
	{
	sum = sum + (*p);
	p++;
	}
	mean = sum / (float) n;
	
	// Compute Variance
	p = a;
	for(i=0;i<n;i++)
	{
	var = var + pow((*p - mean),2);
	p++;
	}
	var = var / (float) n;
	sd = sqrt(var);
	printf("Sum = %f\n", sum);
	printf("Mean = %f\n", mean);
	printf("Standard Deviation = %f\n", sd);
	return 0;
	}
	
	/*
	Sample Output 1:
	$ cc prog11.c -lm
	$ ./a.out
	Enter Number of elements:5
	Enter 5 numbers :1
	2
	3
	4
	5
	Sum = 15.000000
		[10]

Mean = 3.000000
Standard Deviation = 1.414214

[5]

6.a) Define pointers.Explain passing parameters to functions using pointers. [2+3m]

Definition [1]+Program [3]+output [1]

A **pointer** in programming is a variable that stores the memory address of another variable.

The general syntax:data_type *ptr_name;
Example: int *p;

```
#include<stdio.h>
void sum ( int *a, int *b, int *t);
int main()
{
int num1, num2, total;
printf("\n Enter two numbers : ");
scanf("%d,%d", &num1, &num2);
sum(&num1, &num2, &total);
printf("\n Total = %d", total);
return 0;
}
void sum ( int *a, int *b, int *t)
{
*t = *a + *b;
return;
}
```

Output:
Enter two numbers : 2,3
Total = 5

6.b) Differentiate Null pointer and Generic pointer. [5*1M] [5]

Null Pointer	Void Pointer
A pointer that points to no valid memory	A pointer that can point to any data type
It holds no valid address (often NULL or nullptr)	It holds the address of some object, but with no type information
Cannot be dereferenced, as it points to nothing	Can be dereferenced, but must first be cast to the correct type
Indicates absence of an object or uninitialized pointer	Used for generic or flexible pointer references
Not associated with any type	Can point to any type of data

7.	Implement structures to read, write and compute average- marks of the students, list the students scoring above and below the average marks for a class of N students.
	Program [9]+output [1]
	
	struct student
	{
	int id;
	char name[20];
	float sub[6];
	float avg;
	};
	
	int main()
	{
	struct student s[20];
	float sum=0;
	int i,j,n;
	printf("Enter the number of records :");
	scanf("%d",&n);
	printf("Enter %d student details...\n",n);
	for(i=0;i<n;i++)
	{
	printf("\n\nEnter student ID, name :");
	scanf("%d%s",&s[i].id, s[i].name);
	printf("Enter 6 subject marks :");
	
	for (j=0;j<6;j++)
	{
	scanf("%f", &s[i].sub[j]);
	}
	}
	
	for(i=0;i<n;i++)
	{
	sum=0;
	for (j=0;j<6;j++)
	{
	sum = sum + s[i].sub[j];
	}
	s[i].avg = sum / 6;
		[10]
		
		[10]

8.	}
	printf("Students scoring above the average marks....\n");
	printf("\n\nID\tName\tAverage\n\n");
	
	for(i=0;i<n;i++)
	{
	if(s[i].avg>=35.0)
	printf("%d\t%s\t%f\n",s[i].id,s[i].name,s[i].avg);
	}
	printf("\n\nStudents scoring below the average marks....\n");
	printf("\n\nID\tName\tAverage\n\n");
	for(i=0;i<n;i++)
	{
	if(s[i].avg<35.0)
	printf("%d\t%s\t%f\n",s[i].id,s[i].name,s[i].avg);
	}
	return 0;
	}
	Define string. List out any 5 string manipulation built-in functions with examples.
	Definition[1] + [2*5m]
	
	A **string** is a sequence of characters or a data structure used to represent text.
	char str[] = "string_value";
	
	1. strlen()
	Purpose: Returns the length of a string
	
	#include <stdio.h>
	
	#include <string.h>
	
	int main() {
	
	char str[] = "Hello, World!";
	
	printf("Length of the string: %lu\n", strlen(str));
	
	return 0;
	
	}
	
	2. strcpy()
	Purpose: Copies the contents of one string to another.
	#include <stdio.h>
	#include <string.h>
	
	int main() {
	char src[] = "Hello, World!";
	char dest[50];
	
	strcpy(dest, src);
	printf("Destination string: %s\n", dest);
	return 0;
	}
	
	3. strcat()
	Purpose: Concatenates (appends) one string to the end of another.
	#include <stdio.h>
		[10]

```
#include <string.h>

int main() {
    char str1[50] = "Hello, ";
    char str2[] = "World!";

    strcat(str1, str2);  // Append str2 to str1
    printf("Concatenated string: %s\n", str1);  // Output: Hello, World!

    return 0;
}
```

4. strcmp()

Purpose: Compares two strings lexicographically. Returns 0 if the strings are equal, a positive value if the first string is greater, and a negative value if the second string is greater.

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

int main() {
    char str1[] = "apple";
    char str2[] = "banana";

    int result = strcmp(str1, str2);

    if (result == 0) {
        printf("Strings are equal.\n");
    } else if (result < 0) {
        printf("'%s' is less than '%s'.\n", str1, str2);
    } else {
        printf("'%s' is greater than '%s'.\n", str1, str2);
    }

    return 0;
}
```

5. strchr()

Purpose: Searches for the first occurrence of a character in a string.

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

int main() {
    char str[] = "Hello, World!";

    char *result = strchr(str, 'W');  // Find first occurrence of 'W'
    if (result != NULL) {
        printf("Found character '%c' at position: %ld\n", *result, result - str);
    } else {
        printf("Character not found.\n");
    }

    return 0;
}
```