Solution with scheme-Model Answer

Prof.Lynsha Helena Pratheeba HP/Dr.Vijayalakshmi									
Internal Assessment Test 2 – May 2024									
Sub	**Principles of Programming Using C**					Sub code	**BPOPS203**	Branch	**CSE, CSE(DS)**
Date	**22.05.2024**	Duration	**90 mins**	Max Marks	**50**	Sem /Sec	**II Sem P-Cycle (I,J,K,L)**		**OBE**

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1.	Define function with syntax and example. Also explain the various types of parameter passing techniques with example.	[10]	CO3	L2
2. a. b.	What is a recursive function? Write the code for finding the factorial of a number using the same. Differentiate actual and formal parameters.	[6] [4]	CO3	L2
3.	Write a C program to implement bubble Sort technique (ascending order). Demonstrate the steps for the input: 5, 1, 4, 2, 8.	[10]	CO3	L2
4.	Write a C code to implement Matrix multiplication and validate the rules of multiplication.	[10]	CO3	L3
5. a. b.	List out the four different ways to initialize the strings each with example. Define string. Write a program to get a string data from the user and display it using gets and puts function.	[4] [6]	CO4 CO4	L3 L2
6.	Write a C program to create user defined functions that implements string operations such as compare, concatenate, copy and length of the string using parameter passing techniques.	[10]	CO4	L3
7.	Explain Pointers with definition, initialization and pointer arithmetic with example code.	[10]	CO4	L3
8. a. b.	Write a C code to find the area of rectangle by passing pointers to a function. Write a C code to find the larger of two numbers by using function pointer.	[5] [5]	CO4 CO4	L3 L3

1. **Define function with syntax and example. Also explain the various types of parameter passing techniques with example.** [10]

Definition and Syntax [4M]

Function is a building block that contains set of programming statements to perform a particular task.

Function Declaration Syntax:

return-type function-name(datatype1 parameter1, parameter2,................, datatype n parameter n);

Function Call Syntax:

function-name (parameter1,parameter2, ...,parameter n);

Function Definition Syntax:

```
return-type function-name(datatype1 parameter1, parameter2,................, datatype n parameter n)
{
Statements;
}
```

The various types of parameter passing techniques are Call by Value and Call by Reference.

Program with Output [3M]

Call By Value: In this parameter passing method, values of actual parameters are copied to function's formal parameters and the two types of parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of the caller.

Example:

```
#include<stdio.h>
void swapx(int x, int y);
int main()
{
int a = 10, b = 20;
swapx(a, b);
printf("a=%d b=%d\n", a, b);
return0;
}
void swapx(intx,inty)
{ int t;
t = x;
x = y;
y = t;
printf("x=%d y=%d\n", x, y);
}
```

Sample Output:

```
x=20 y=10
a=10 b=20
```

Program with Output [3M]

Call by Reference: Both the actual and formal parameters refer to the same locations, so any changes made inside the function are actually reflected in actual parameters of the caller.

Example:

```
#include <stdio.h>
voidswapx(int*,int*);
int main()
{
int a = 10, b = 20;
swapx(&a,&b);
printf("a=%d b=%d\n", a, b);
return0;
 }
void swapx(int*x,int*y)
{ intt;
t =*x;
*x =*y;
*y =t;
printf("x=%d y=%d\n", *x, *y);
}
```

2 a **What is a recursive function? Write the code for finding the factorial of a number using the same.** [6]

Definition and Syntax [2M]

Recursive function can be defined as a routine that calls itself the function directly or indirectly.

Syntax Model:

```
int recursion(x);
{
if(x==0)
return;
recursion(x-1);}
```

Program with Output [4M]

```
#include<stdio.h>
int find_factorial(int);
int main()
{
intnum,fact;
printf("\nEnter any integer number:");
scanf("%d",&num);
fact=find_factorial(num);
printf(" \nfactorial of %d is: %d",num, fact);
return0;
}
int find_factorial(int n) {
//Factorial of 0 is 1
if(n==0)
return(1);
return(n*find_factorial(n-1));
}
```

Output:

```
Enter any integer number:5
factorial of 5 is:120
```

2 b **Differentiate actual and formal parameters.** [4]

Differences [each 1M]

Actual Parameters	Formal Parameters
When a function is called, the values (expressions) that are passed in the function call are called the arguments or actual parameters.	The parameter used in function definition statements which contain data type on its time of declaration is called formal parameter.
These are the variables or expressions referenced in the parameter list of a subprogram call.	These are the variables or expressions referenced in the parameter list of a subprogram specification.
Actual Parameters are the parameters which are in calling subprogram.	Formal Parameters are the parameters which are in called subprogram.
There is no need to specify datatype in actual parameter.	The datatype of the receiving value must be defined.

3. **Write a C program to implement bubble Sort technique (ascending order). Demonstrate the steps for the input: 5, 1, 4, 2, 8.** [10]

Program [4M]

Program:

```
#include<stdio.h>
int main()
{
int a[20],n,i,j,temp;
printf("Enter the number of elements");
scanf("%",&n);
printf("Enter %d integers);
for(i=0;i<n;i++)
{
Scanf("%d",&a[i]);
}
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;
}}}
printf(”The sorted array is ....\n;);
for(i=0;i<n;i++)
{
Printf(”%d”,a[i]);
}
Printf("\n");
return 0;
}
```

Output [3]

Output

```
Enter the number of elements:5
Enter 5 integers: 5, 1, 4, 2, 8
The sorted array is 1,2,4,5,8
```

Steps[3]

Steps:

Iteration 1:
Compare 5 and 1, swap -> [1, 5, 4, 2, 8]
Compare 5 and 4, swap -> [1, 4, 5, 2, 8]
Compare 5 and 2, swap -> [1, 4, 2, 5, 8]
Compare 5 and 8, no swap -> [1, 4, 2, 5, 8]
After Iteration 1: 1 4 2 5 8
Iteration 2:
Compare 1 and 4, no swap -> [1, 4, 2, 5, 8]
Compare 4 and 2, swap -> [1, 2, 4, 5, 8]
Compare 4 and 5, no swap -> [1, 2, 4, 5, 8]
Compare 5 and 8, no swap -> [1, 2, 4, 5, 8]
After Iteration 2: 1 2 4 5 8
Iteration 3:
Compare 1 and 2, no swap -> [1, 2, 4, 5, 8]
Compare 2 and 4, no swap -> [1, 2, 4, 5, 8]
Compare 4 and 5, no swap -> [1, 2, 4, 5, 8]
Compare 5 and 8, no swap -> [1, 2, 4, 5, 8]
After Iteration 3: 1 2 4 5 8
Iteration 4:
Compare 1 and 2, no swap -> [1, 2, 4, 5, 8]
Compare 2 and 4, no swap -> [1, 2, 4, 5, 8]
Compare 4 and 5, no swap -> [1, 2, 4, 5, 8]
Compare 5 and 8, no swap -> [1, 2, 4, 5, 8]
After Iteration 4: 1 2 4 5 8

4 **Write a C code to implement Matrix multiplication and validate the rules of multiplication.** [10]

Program [7]

Program:

```
#include<stdio.h>
int main()
{
int a[10][10],b[10][10],c[10][10];
int m,n,p,q;
int i,j,k;
printf("Enter the order of matrix A :");
scanf("%d%d",&m,&n);
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("Multiplication of matrices not possible....\n");
return (-1);
}
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]);
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");
}
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]);
printf("\nThe matrix B is ---\n");
for(i=0;i<p;i++)
{
for(j=0;j<q;j++)
{
printf("%d\t",b[i][j]);
}
printf("\n");
}
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]);
}
}
}
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[3]

Output:

```
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
```

5 a **List out the four different ways to initialize the strings each with example.** [4]

Types with example [4M]

1. Assigning a String Literal without Size
   Eg: char str[] = "CMRIT";
2. Assigning a String Literal with a Predefined Size
   Eg: char str[7] = "CMRIT";
3. Assigning Character by Character with Size
   Eg: char str[7] = { 'C','M','R','I','T','\0'};
4. Assigning Character by Character without Size
   Eg: char str[] = { 'C','M','R','I','T','\0'};

5 b **Define string. Write a program to get a string data from the user and display it using gets and puts function.**

Definition[2]

Strings are defined as an array of characters. The difference between a character array and a string is

the string is terminated with a special character ‘\0’.
Eg: char c[]=”Hi Students” [6]

Program[4]

Program:
```
int main()
{
char str[100];
printf("Enter a string: ");
fgets(str, sizeof(str), stdin);
str[strcspn(str, "\n")] = 0;
printf("You entered: ");
puts(str);
return 0;
}
```
Output:
```
Enter a string: Hello
You entered: Hello
```

6. **Write a C program to create user defined functions that implements string operations such as compare, concatenate, copy and length of the string using parameter passing techniques.** [10]

Program[7M]

Program:
```
 #include <stdio.h>
  int str_length(char []);
 int str_compare(char [], char []);
 void str_concat(char [], char []);
  int main()
 {
 char str[50];
 char str1[50], str2[50];
 char str_des[100], str_src[50];
 int length, comp_res;
 printf("\nEnter a string :");
 scanf("%s",str);
 length = str_length(str);
 printf("The length of %s is %d\n",str,length);
 printf("\nEnter two strings for string compare :");
 scanf("%s%s",str1,str2);
 comp_res=str_compare(str1,str2);
 if (comp_res < 0)
 {
 printf("String \"%s\" is less than string \"%s\"\n",str1,str2);
 }
 else if (comp_res == 0)
 {
 printf("String \"%s\" is same as string \"%s\"\n",str1,str2);
 }
 else
 {
 printf("String \"%s\" is greater than string \"%s\"\n", str1,str2);
 }
 printf("\nEnter two strings for string concatenation :");
 scanf("%s%s",str_des,str_src);
 str_concat(str_des,str_src);
 printf("The string after concatenation is \"%s\"\n", str_des);
 return 1;
 }
 int str_length(char s[])
 {
 int i;
for(i=0;s[i]!='\0';i++);
return i;
 }
 int str_compare(char s1[], char s2[])
 {
int i,j;
for(i=0,j=0;(s1[i] != '\0' && s2[j] != '\0');i++,j++)
 {
 if (s1[i] != s2[j])
 {
 return (s1[i] - s2[j]);
 }
 }
 if (s1[i] == '\0' && s2[j] == '\0')
 {
 return 0;
}
 else if(s1[i] == '\0' || s2[i] == '\0')
 {
 return (s1[i] - s2[i]);
 }
 }
void str_concat(char s1[], char s2[])
 {
 int i,j;
for(i=0;s1[i] != '\0';i++);
for(j=0;s2[j] != '\0';i++,j++)
 {
 s1[i] = s2[j];
 }
 s1[i] = '\0';
 }
```

Output[3M]

Output

```
Enter a string :rainbow
The length of rainbow is 7
Enter two strings for string compare :rain
rainbow
String "rain" is less than string "rainbow"
Enter two strings for string concatenation :rain
bow
The string after concatenation is "rainbow"
```

7. **Explain Pointers with definition, initialization and pointer arithmetic with example code.**

Definition and Syntax[4]

A pointer in C is a variable that stores the memory address of another variable. Pointers provide a way to directly access and manipulate memory, allowing for efficient handling of arrays, dynamic memory allocation, and complex data structures.

Syntax for Pointer declaration [10]

datatype *ptr;
tr = &var;

Increment Decrement, pointer addition, pointer subtraction(6)

8 a **Write a C code to find the area of rectangle by passing pointers to a function.** [5]

Program[4.5]

```
Program:
#include <stdio.h>
void calculateArea(int *length, int *width, int *area);
int main() {
int length, width, area;
printf("Enter the length of the rectangle: ");
scanf("%d", &length);
printf("Enter the width of the rectangle: ");
scanf("%d", &width);
calculateArea(&length, &width, &area);
printf("The area of the rectangle is: %d\n", area);
return 0;
}
void calculateArea(int *length, int *width, int *area) {
*area = (*length) * (*width);
}
```

Output[0.5]

```
Output:
Enter the length of the rectangle: 5
Enter the width of the rectangle: 10
The area of the rectangle is: 50
```

8 b **Write a C code to find the larger of two numbers by using function pointer.** [5]

Program[4.5]

```
Program:
#include <stdio.h>
int findLarger(int a, int b);
int main() {
int num1, num2, larger;
int (*funcPtr)(int, int); // Declaration of function pointer
Assign the function pointer to the findLarger function
funcPtr = findLarger;
printf("Enter the first number: ");
scanf("%d", &num1);
printf("Enter the second number: ");
canf("%d", &num2);
larger = funcPtr(num1, num2);
printf("The larger number is: %d\n", larger);

return 0;
}
int findLarger(int a, int b) {
return (a > b) ? a : b;
}
```

Output[0.5]

	Output: Enter the first number: 25 Enter the second number: 10 The larger number is: 25	