CMR INSTITUTE OF TECHNOLOGY

Internal Assessment Test 2 – May 2017-SCHEME & SOLUTION

Sub:	Programming in C and Data Structures							Code:	15PCD23
Date:	09/ 05/2017	Duration:	90mins	Max Marks:	50	Sem:	II	Branch:	B,C,D,E Section

Note: Answer any five questions:

		Marks	CO	RBT
1	**a) Explain any 4 string manipulation functions with syntax.**			
	strlen() • This function calculates the length of string. It takes only one argument, i.e., string-name. • The syntax is shown below: temp_variable = strlen(string_name); **strcpy()** • This function copies the content of one string to the content of another string. It takes 2 arguments. • The syntax is shown below: strcpy(destination,source); where source and destination are both the name of the string. **strcat()** • This function joins 2 strings. It takes two arguments, i.e., 2 strings and resultant string is stored in the first string specified in the argument. • The syntax is shown below: strcat(first_string,second_string); **strcmp()** • This function compares 2 string and returns value 0, if the 2 strings are equal. It takes 2 arguments, i.e., name of two string to compare. • The syntax is shown below: temp_varaible=strcmp(string1,string2); • If string1 is greater than string 2: +1 is returned by the function. • If string1 is lesser than string 2: -1 is returned by the function.	04	CO 4	L1
	b) Write and execute a C program that implements string copy operation *STRCOPY*(str1,str2) that copies a string *str1* to another string *str2* without using library function *strcpy ().*	06	CO 4	L3

```
#include<stdio.h>
// Function declaration for strcopy.
void strcopy ( char str1 [ ], char str2 [ ] );
int main()
{
char str1 [ 20 ], str2 [ 20 ];
// Input the string that you want to copy.
printf ( "\nEnter string to copy: " );
gets ( str1 );
// Function call.
```

```
strcopy ( str1 , str2 );
// Display the contents of str1 and str2.
printf ( "\n\nCopying success!!!!!\n" );
printf ( "\nThe first string is: " );
puts ( str1 );
printf ( "\nThe second string is: " );
    puts ( str2 );

return 0;
}
// Function definition for strcopy.
void strcopy ( char str1 [ ], char str2 [ ] )
{
int i;
// Copying the contents of str1 to str2 until NULL is encountered.
i = 0;
while ( str1 [ i ] != '\0' )
{
str2 [ i ] = str1 [ i ];
i++;
}
// Append NULL character at the end of str2.
str2 [ i ] = '\0';
}
```

2 a) Define Array. Explain declaration and initialization of 2D array with syntax and example. — **05** — **CO 4** — **L2**

ARRAY

- Array is a collection of elements of same data type.
- The elements are stored sequentially one after the other in memory.
- Any element can be accessed by using
  → name of the array
  → position of element in the array
- Arrays are of 2 types:

1) Single dimensional array
2) Multi dimensional array

2 DIMENSIONAL ARRAY

- A 2 dimensional array is a matrix consisting of related elements of same type.
- In memory, all the elements are stored in continuous memory-location one after the other.

Declaration of 2 Dimensional Arrays

- The syntax is shown below:

```
data_type array_name[row_size][Col_size];
```

where data_type can be int, float or char
array i_name s name of the array
row_size indicates number of rows and col_size indicates number of cols in the array

- For ex:

```
int matrix[3][3];
```

Initialization of 2 Dimensional Arrays

For ex:

	int matrix[3][3]={{2,4,4},{1,1,1},{2,2,2}}; or for(i=0;i<3i++) { for(j=0; j<3; j++) { scanf("%d",&matrix[i][j]); } } **b) Develop an algorithm, implement and execute aC program that reads N integerNumbers and arrange them in ascending order using Bubble Sort.** **Input:** An array of integers entered in random order. **Output:** To sort the array in ascending order using Bubble Sort. **Step 1: Start** **Step 2:** Read the number of integers: **n** **Step 3:** Read the numbers: **a [n]** **Step 4:** Initialize: **i = 0** **j = 0** **Step 5:** Check **if i is less than n - 1.** If true **goto Step 6** otherwise **goto Step 11** **Step 6:** Check **if j is less than n - 1 - i**. If true **goto Step 7** otherwise **goto Step 10** **Step 7:** Check **if a [j] is greater than a [j + 1]**. If true **goto Step 8** otherwise **goto Step 9** **Step 8:** Compute: **temp = a [j]** **a [j] = a [j + 1]** **a [j + 1] = temp** **Step 9: Increment j goto Step 6** **Step 10: Increment i goto Step 5** **Step 11: Display the sorted array using Bubble Sort: a [n]** **Step 12: Stop** #include<stdio.h> int main() { int a [25], i, j, n, temp; **// Input the number of integers that you want to sort.** printf ("\nEnter the number of integers to be sorted: "); scanf ("%d" , &n); **// Input the numbers.** printf ("\nEnter the numbers:\n"); for (i = 0 ; i < n ; i++) { scanf ("%d" , &a [i]); } **// Bubble Sort.** for (i = 0 ; i < n - 1 ; i++) **// Passes.** { for (j = 0 ; j < n - 1 - i ; j++) **// Comparisions.** {	**05**	**CO 4**	**L3**

```
if ( a [ j ] > a [ j + 1 ] ) // If jth element is greater than j+1th element then swap.
{
temp = a [ j ];
a [ j ] = a [ j + 1 ];
a [ j + 1 ] = temp;
}
}
}
// Display sorted array in ascending order.
printf ( "\n\nThe sorted array using Bubble Sort is:\n" );
for ( i = 0 ; i < n ; i++ )
{
printf ( "%d\t", a [ i ] );
}
printf ( "\n\n" );
return 0;
}
```

3	**a) Develop, implement and execute a C program to search a Name in a list of names using Binary searching Technique.**	**06**	**CO 4**	**L3**

```
#include<stdio.h>
#include<string.h>
    int main()

{
char s [ 20 ] [ 20 ], temp [ 20 ], search_string [ 20 ];
int n, i, j, first, last, mid, res;
//Input the number of strings.
printf ( "\nEnter the number of strings: " );
scanf ( "%d", &n );
//Input the strings.
printf ( "\nEnter the strings in sorted order:\n" ) ;
for ( i = 0 ; i < n ; i++ )
{
scanf ( "%s", s [ i ] );
}
// Input the string that you want to search.
printf ( "\nEnter string to search: " );
scanf ( "%s", search_string );
// Search for string using Binary Search.
first = 0;
last = n-1;
while ( first <= last )
{
mid = ( first + last ) / 2;
res = strcmp ( s [ mid ], search_string );
// The string in mid position and search string are same.
if ( res == 0 )
{
printf ( "\nString found at %d position\n\n", mid );
return 0;
}
// The string in mid position is greater than search string.
// Search in the first half of the array.
```

```
else if ( res > 0 )
{
last = mid - 1;
}
// The string in mid position is less than search string.
// Search in the second half of the array.
else
{
first = mid + 1;
}
}
printf ( "\nString not found\n\n" );
return 0;
}
```

	b) Explain the scope of global and local variable with example. • Variables that are declared inside a function or block are called local variables. #include <stdio.h> int main () { /* local variable declaration */ int a, b; int c; /* actual initialization */ a = 10; b = 20; c = a + b; printf ("value of a = %d, b = %d and c = %d\n", a, b, c); return 0; } • Global variables are defined outside a function, usually on top of the program. #include <stdio.h> /* global variable declaration */ int g; int main () { /* local variable declaration */ int a, b; /* actual initialization */ a = 10; b = 20; g = a + b; printf ("value of a = %d, b = %d and g = %d\n", a, b, g); return 0; }	**04**	**CO 3**	**L2**

4	a)	**Define a function (includes function definition, call and prototype declaration).** A **function** is a group of statements that together perform a task. Every **C** program has at least one **function**, which is main(), and all the most trivial programs can define additional **functions**. **FUNCTION DECLARATION:** Syntax: <return type> function_name(Parameter list….); Parameter list contains <data type> var_name separately for every parameter. **FUNCTION DEFINITION:** Syntax: <return type> function_name(Parameter list….) { ---------- ---------- ---------- **}** **FUNCTION CALL** Syntax: function_name(parameters**);**	**06**	**CO 3**	**L3**
	b)	**Explain with example how to pass array(s) to a function.** If you want to pass a single-dimension array as an argument in a function, you would have to declare a formal parameter in one of following three ways and all three declaration methods produce similar results because each tells the compiler that an integer pointer is going to be received. Similarly, you can pass multi-dimensional arrays as formal parameters. Way-1 Formal parameters as a sized array − `void myFunction(int param[10]){` . . . `}` Way-2 Formal parameters as an unsized array − `void myFunction(int param[]){` . . . `}` `#include <stdio.h>`	**04**	**CO 3**	**L3**

```
/* function declaration */
double getAverage(int arr[], int size);

int main () {

  /* an int array with 5 elements */
  int balance[5] = {1000, 2, 3, 17, 50};
  double avg;

  /* pass pointer to the array as an argument */
  avg = getAverage( balance, 5 ) ;

  /* output the returned value */
  printf( "Average value is: %f ", avg );

  return 0;
}
double getAverage(int arr[], int size) {

  int i;
  double avg;
  double sum = 0;

  for (i = 0; i < size; ++i) {
    sum += arr[i];
  }

  avg = sum / size;

  return avg;
}
```

5	**a) Explain two mechanisms of argument passing example.** **ARGUMENT PASSING – CALL BY VALUE** • In this type, value of actual arguments are passed to the formal arguments and the operation is done on the formal arguments. • Any changes made in the formal arguments does not effect the actual arguments because formal arguments are photocopy of actual arguments. • Changes made in the formal arguments are local to the block of called-function. • Once control returns back to the calling-function the changes made vanish. **CALL BY REFERENCE** When, argument is passed using pointer, address of the memory-location is passed instead of value. So all the changes are reflected outside the function definition. When we will swap the value of the two variables, output will beVariables with changed values.	**06**	**CO 3**	**L2**
	b) Design and develop a C function isprime(num) that accepts an integer argument and returns 1 if the argument is prime, a 0 otherwise. Write a C program that	**04**	**CO 3**	**L3**

	invokes this function to generate prime numbers between the given range. #include<stdio.h> **// Function declaration.** int prime (int n); int main() { int n, n1, n2, i, range; **// Input a number that you want to check for prime.** printf ("\nEnter a positive integer to test for primality: "); scanf ("%d", &n); **// Check if the entered number is positive or not.** if (n <= 0) { printf ("\nError: zero or -ve integers by definition cannot be prime!\n"); return 1; } **// Input the range to generate prime numbers.** printf("\nEnter two numbers between which the prime numbers has to be generated: "); scanf("%d%d", &n1, &n2); **// Display prime numbers between the specified range.** printf("\nPrime numbers between %d and %d are :\n", n1, n2); for (i = n1+1; i < n2; i++) { range = prime (i); if (range == 1) printf("%d\t",i); } printf("\n"); return 0; } **// Function definition.** int prime (int n) { int i; for (i = 2 ; i <= n/2 ; i++) { if (n % i == 0) { return 0; **// if i divides n then NOT Prime number.** } } return 1; **// Prime number.** }			
6	**a) What is recursion? Write a C program to calculate factorial of a number using recursive function.** Recursion is a programming technique that allows the programmer to express operations in terms of themselves. In C, this takes the form of a function that calls itself. **// Function declaration.** long int fact (long int n);	**06**	**CO 3**	**L3**

	int main() { long int n, factorial; **// Input the values of n.** printf ("\nEnter the value of n : "); scanf ("%ld", &n); if(n<0) { printf ("\nError!!! n not be -ve\n\n"); return 0; } else factorial = fact (n); printf ("\nFactorial of %ld is: %ld\n", n, factorial); return 0; **}**			
	b) Write a programto reverse a sentence using recursion. `#include<stdio.h>` `#include<string.h>` `/* function STR_reverse(char *, int)---> takes the string s and reverse it.*/` `void STR_reverse( char *s, int len)` `{` `if(len<=0)` `return;` `char temp;` `temp = s[0];` `s[0] = s[len-1];` `s[len-1] = temp;` `STR_reverse(s+1, len-2);` `}` `void main()` `{` `char s[50];` `printf("Enter the string: ");` `gets(s); // input the string` `printf("The original string is: %s\n", s);//display the original string before it is` `/* Next we will call STR_reverse()*/` `STR_reverse(s, strlen(s));` `printf("The reverse of the string is: %s\n", s);// display the reverse` `}`	**04**	**CO 3**	**L3**
7	**a) What is a structure? Explain structure declaration withexample.** "A structure is a used-defined data type, which contains group of multiple different data typerelated variable." We can define the structure by following format: struct tag_name { datatype element_1;	**04**	**CO 5**	**L2**

	datatype element_2; … datatypeelement_n; }; Structure variable can be declared as following way: struct tag_name variable_1, variable_2, ⋯ , variable_n; e.g. struct student { // structure template int rollno; char name[100]; char address[100]; }; struct student s1,s2,s3; // structure declaration **b) Explain with example how to pass structure variable(s) as function argument(s).** • A structure can be passed to any function from main function or from any sub function. • Structure definition will be available within the function only. • It won't be available to other functions unless it is passed to those functions by value or by address(reference). • Else, we have to declare structure variable as global variable. That means, structure variable should be declared outside the main function. So, this structure will be visible to all the functions in a C program. **PASSING STRUCTURE TO FUNCTION IN C:** It can be done in below 2 ways. 1. Passing structure to a function by value 2. Passing structure to a function by address(reference) **EXAMPLE PROGRAM – PASSING STRUCTURE TO FUNCTION IN C BY VALUE:** #include <stdio.h> #include <string.h> structstudent { intid; charname[20]; floatpercentage; }; voidfunc(structstudent record); intmain() { structstudent record; record.id=1; strcpy(record.name,"Raju"); record.percentage=86.5; func(record); return0; } voidfunc(structstudent record) { printf(" Id is: %d \n",record.id); printf(" Name is: %s \n",record.name); printf(" Percentage is: %f \n",record.percentage); }	**06**	**CO 5**	**L3**

EXAMPLE PROGRAM – PASSING STRUCTURE TO FUNCTION IN C BY ADDRESS:

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

structstudent
{
        intid;
        charname[20];
        floatpercentage;
};

voidfunc(structstudent*record);

intmain()
{
       structstudent record;

       record.id=1;
       strcpy(record.name,"Raju");
       record.percentage=86.5;

       func(&record);
       return0;
}

voidfunc(structstudent*record)
{
       printf(" Id is: %d \n",record->id);
       printf(" Name is: %s \n",record->name);
       printf(" Percentage is: %f \n",record->percentage);
}
```

8	**a) Write a C program to maintain a record of n book details using an array of structures with four fields (Book name, Author name, ISBN number, and price). Assume appropriate data type for each field. Print the details of the book, given the Book name as input.**	**07**	**CO 5**	**L3**

```
/* Program to create details of a student using structures. */
#include<stdio.h>
#include<string.h>
// Name of defined structure type is student.
struct book
{
int isbn;
char bname [ 30 ],author[30];
float price;
};
struct book b[ 30 ];
int main()
{
int i, n;
     char book_name [ 30 ];

// Input the number of students.
printf ( "\nEnter the number of books: " );
scanf ( "%d" , &n );
// Input student details.
for ( i = 0 ; i < n ; i++ )
{
printf ( "\n\nEnter the following details for Book %d", i );
```

```
printf ( "\nISBN No: " );
    scanf ( "%d" , &b[i].isbn);

printf ( "\nBook Name: " );
scanf ( "%s" , b[i].bname );
printf ( "\nAuthor Name: " );
scanf ( "%s" , b[i].author );
printf ( "\n Price: " );
scanf ( "%f" , &b[i].price);

}
// Input the student name that you want to search for.
printf ( "\n\nEnter the book name you wish to search: " );
    scanf ( "%s" , book_name );

// Linear Search.
for ( i = 0 ; i < n ; i++ )
{
if ( strcmp ( book_name, b[i].bname ) == 0 )
{
printf ( "\nDetailsof the book %s is:\nISBN No.- %d\nAuthor Name-%s\nPrice-
%f\n",book_name,b[i].isbn,b[i].author,b[i].price  );
return 0;
}
}
    // Display record not found

printf ( "\nBook not Found!!!!\n\n" );
return 0;
    }
```

b) What is typedef? Explain with example.

03 | **CO 5** | **L1**

The C programming language provides a keyword called typedef, which you can use to give a type, a new name. Following is an example to define a term BYTE for one-byte numbers

```
typedef unsigned char BYTE;
```

After this type definition, the identifier BYTE can be used as an abbreviation for the type unsigned char, for example..

By convention, uppercase letters are used for these definitions to remind the user that the type name is really a symbolic abbreviation, but you can use lowercase, as follows –

```
typedef unsigned char byte;
```

typedef to give a name to your user defined data types as well.