CMR
INSTITUTE OF
TECHNOLOGY

Improvement Test- Scheme and Solution

Sub:	PROGRAMMING IN C AND DATA STRUCTURES							Code:	15PCD13
Date:	19 / 11 / 2016	Duration:	90 mins	Max Marks:	50	Sem:	I	Branch:	ECE/EEE/TCE/CIV
Answer Any FIVE FULL Questions									

		Marks	OBE	
			CO	RBT
1(a)	**What is iterative function and recursive function? Write a recursive function to find nth series. Explain function prototype, function call and function definition with the help of example.**	[10]	CO4	L2

Iterative functions repeat a set of code until a specified condition is reached and recursive functions call itself until a specified condition is reached.

```
#include<stdio.h>

int Fibonacci(int);

main()
{
  int n, i = 0, c;

  scanf("%d",&n);

  printf("Fibonacci series\n");

  for ( c = 1 ; c <= n ; c++ )
  {
    printf("%d\n", Fibonacci(i));
    i++;
  }

  return 0;
}

int Fibonacci(int n)
{
  if ( n == 0 )
    return 0;
  else if ( n == 1 )
    return 1;
  else
    return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

OR

(b)	**Write a program to perform binary search on list of names. Explain concept of array of string with respect to binary search example.**	[6+4]	CO3	L2

```
/* Program to perform Binary Search on strings. */
#include<stdio.h>
#include<string.h>
int main()
{
        char s [ 20 ] [ 20 ], temp [ 20 ], search[ 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);
        // 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;
                }
                else if ( res > 0 )
                {
                last = mid - 1;
                }
                else
                {
                first = mid + 1;
                }
        }
        printf ( "\nString not found\n\n" );
        return 0;
}
```

2(a)	**What is a Structure? What is typedef? Explain structure within a**	[5+5]	CO6	L3

structure with example.

Structure is a group of elements of different data types. It is used to create user defined data type.

typedef is used create alternate names of datatypes to new names.

Example of structure within structure :

```
#include<stdio.h>
#include<string.h>
void main()
{
        struct student{
        introllno;
        char name[100];
        char grade[10];
        int marks;
        struct dob{
int d;
        int m;
        int y;
        }dob;
}s[100];
        inti,n;
        char key[90];
        printf("enter the no of students in class\n");
        scanf("%d",&n);
        printf("Enter the details of students\n");
        for(i=0;i<n;i++)
                {
                        printf("enter student roll no for student[%d]\n",i);
                        scanf("%d",&s[i].rollno);
                        printf("enter student name for student[%d]\n",i);
                        scanf("%s",s[i].name);
                        printf("enter student grade for student[%d]\n",i);
                        scanf("%s",s[i].grade);
                        printf("enter student marks for student[%d]\n",i);
                        scanf("%d",&s[i].marks);
                        printf("enter student date of birth for student[%d]\n",i);
                        scanf("%d%d%d",&s[i].dob.d,&s[i].dob.m,&s[i].dob.y);
                }
        for(i=0;i<n;i++)
                {
                        printf("student roll no for student[%d] is  %d\n"
,i,s[i].rollno);
                        printf("student name for student[%d] is %s\n",i,s[i].name);
                        printf("student grade for student[%d] is %s\n",i,s[i].grade);
                        printf("student marks for student[%d] is %d\n",i,s[i].marks);
                        printf("student date of birth for student[%d] is %d-%d-
%d\n",i,s[i].dob.d,s[i].dob.m,s[i].dob.y);

                }
```

```
        printf("enter the name to be searched\n");
        scanf("%s",key);
        for(i=0;i<n;i++)
        {
                if(strcmp(s[i].name,key)==0)
                        {
                                printf("the marks are %d\n",s[i].marks);
                                printf("student date of birth for student is %d-%d-
%d",s[i].dob.d,s[i].dob.m,s[i].dob.y);
                        }
        }
}
```

OR

(b) **Write a C program to maintain a record of n student details using an array of structures with four fields (Roll number, Name, Marks, and Grade). Assume appropriate data type for each field. Print the marks of the student, given the student name as input.** [10] CO6 L3

```
/* Program to create details of a student using structures. */
#include<stdio.h>
#include<string.h>
int main()
{
        struct student
        {
                int rollno;
                char name [ 30 ];
                int marks;
                char grade [ 2 ];
        };
        struct student stud_arr [ 30 ];

        int i, num_studs;
        char stud_name [ 30 ];
        // Input the number of students.
        printf ( "\nEnter the number of students: " );
        scanf ( "%d" , &num_studs );
        // Input student details.
        for ( i = 0 ; i < num_studs ; i++ )
        {
                printf ( "\n\nEnter the following details for Student %d", i );
                printf ( "\nRoll No: " );
                scanf ( "%d" , &stud_arr[i].rollno );
                printf ( "\nName: " );
                scanf ( "%s" , stud_arr[i].name );
                printf ( "\nMarks: " );
                scanf ( "%d" , &stud_arr[i].marks );
                printf ( "\nGrade: " );
                scanf ( "%s" , stud_arr[i].grade );
        }
```

```
        // Input the student name that you want to search for.
        printf ( "\n\nEnter the student name you wish to search: " );
        scanf ( "%s" , stud_name );
        // Linear Search.
        for ( i = 0 ; i < num_studs ; i++ )
        {
                if ( strcmp ( stud_name, stud_arr[i].name ) == 0 )
                {
                printf ( "\nMarks obtained by %s is: %d\n\n", stud_arr[i].name,
        stud_arr[i].marks );
                return 0;
                }
        }
        // Display record not found.
        printf ( "\nStudent Record not Found!!!!\n\n" );
        return 0;
}
```

3(a)	**What is a Pointer? Explain initialization of pointer and arrays. Write a program to find and display sum of n numbers using pointers.**	[10]	CO6	L3

A pointer is a variable which holds address of another variable or a memory-location.
• For ex:
c=300;
pc=&c;
Here pc is a pointer; it can hold the address of variable c
& is called reference operator
Dereference operator(*) are used for defining pointer-variable.
• The syntax is shown below:
data_type *ptr_var_name;
• For ex:
int *a; // a as pointer variable of type int
float *c; // c as pointer variable of type float

Consider an array:
int arr[4];
• The above code can be pictorially represented as shown below:
• The name of the array always points to the first element of an array.
• Here, address of first element of an array is &arr[0].
• Also, arr represents the address of the pointer where it is pointing. Hence, &arr[0] is equivalent to
arr.
• Also, value inside the address &arr[0] and address arr are equal. Value in address &arr[0] is arr[0]
and value in address arr is *arr. Hence, arr[0] is equivalent to *arr.
• Similarly,
&a[1] is equivalent to (a+1) AND, a[1] is equivalent to *(a+1).
&a[2] is equivalent to (a+2) AND, a[2] is equivalent to *(a+2).
&a[3] is equivalent to (a+1) AND, a[3] is equivalent to *(a+3).
.
.
&a[i] is equivalent to (a+i) AND, a[i] is equivalent to *(a+i).
• You can declare an array and can use pointer to alter the data of an array.

• Example: Program to access elements of an array using pointer.

```
#include<stdio.h>
void main()
{
int data[5], i;
printf("Enter elements: ");
for(i=0;i<5;++i)
scanf("%d", data+i );
printf("You entered: ");
for(i=0;i<5;++i)
printf("%d ", *(data+i) );
}
```

Output:

Enter elements: 1 2 3 5 4
You entered: 1 2 3 5 4

• Example: Program to find sum of all the elements of an array using pointers.

```
#include<stdio.h>
void main()
{
int i, a[10], n, sum=0;
printf("Enter the size of the array:");
scanf("%d", &n);
printf("Enter the elements into the array: ");
for(i=0;i<n;i++)
scanf("%d ",&a[i]);
for(i=0;i<n;i++)
sum+=*(a+i);
printf("Sum --> %d ",sum);
 }
```

OR

(b) **What is Dynamic memory allocation? Explain different Dynamic memory allocation function in C with syntax and examples.** [5+5] CO5 L2

The process of allocating memory during program execution is called dynamic memory allocation.C language offers 4 dynamic memory allocation functions. They are,

1. malloc()
2. calloc()
3. realloc()
4. free()

S.no	Function	Syntax
i	malloc ()	malloc (number *sizeof(int));
ii	calloc ()	calloc (number, sizeof(int));
iii	realloc ()	realloc (pointer_name, number * sizeof(int));
iv	free ()	free (pointer_name);

i. malloc() function in C:

- malloc () function is used to allocate space in memory during the execution of the program.
- malloc () does not initialize the memory allocated during execution. It carries garbage value.

- malloc () function returns null pointer if it couldn't able to allocate requested amount of memory.

Example program for malloc() function in C:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
char *mem_allocation;
/* memory is allocated dynamically */
mem_allocation = malloc( 20 * sizeof(char) );
if( mem_allocation== NULL )
{
printf("Couldn't able to allocate requested memory\n");
}
else
{
strcpy( mem_allocation,"CMRIT.com");
}
printf("Dynamically allocated memory content : " \ n%s\n", mem_allocation );
free(mem_allocation);
}
```

Output: Dynamically allocated memory content CMRIT .com

ii. calloc() function in C:

 calloc () function is also like malloc () function. But calloc () initializes the allocated memory to zero. But, malloc() doesn't.

Example program for calloc() function in C:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
char *mem_allocation;
/* memory is allocated dynamically */
mem_allocation = calloc( 20, sizeof(char) );
if( mem_allocation== NULL )
{
printf("Couldn't able to allocate requested memory\n");
}
else
```

```
{
strcpy( mem_allocation," CMRIT.com");
}
printf("Dynamically allocated memory content : " \ "%s\n", mem_allocation );
free(mem_allocation);
}
```

Output: Dynamically allocated memory content
: CMRIT.com

iii. realloc() function in C:

- realloc () function modifies the allocated memory size by malloc () and calloc () functions to new size.
- If enough space doesn't exist in memory of current block to extend, new block is allocated for the full size of reallocation, then copies the existing data to new block and then frees the old block.

iv. free() function in C:

- free () function frees the allocated memory by malloc (), calloc (), realloc () functions and returns the memory to the system.

4(a) **What are preprocessor directives? Explain any five preprocessor directives in C.** [2+8] CO4 L2

The C Preprocessor is not part of the compiler, but is a separate step in the compilation process. In simplistic terms, a C Preprocessor is just a text substitution tool.

i) **#include** - This directive allows external header files to be processed by the compiler.

Syntax:

#include <*header-file***> or #include "***source-file***"**

When enclosing the file with < and >, then the implementation searches the known header directories for the file (which is implementation-defined) and processes it. When enclosed with double quotation marks, then the entire contents of the source-file is replaced at this point. The searching manner for the file is implementation-specific.

Examples:

#include <stdio.h> #include "my_header.h"

ii)#pragma- The #pragma directive allows a directive to be defined. Its effects are implementation-defined. If the pragma is not supported, then it is ignored.

Syntax: #pragma ***directive***

- #pragma page( ) // Forces a form feed in the listing
- #pragma warning(disable: 2044) // Disables warning number 2044
- #pragma line 100 // Sets the current line number to 100 for listing and reporting purposes.

OR

(b)	**Write note on a) primitive and non-primitive data type b) Abstract data type.**	[6+4]	CO6	L1

Primitive Data Type:

The primitive data types are the basic data types that are available in most of the programming languages. The primitive data types are used to represent single values.

- **Integer:** This is used to represent a number without decimal point.

Eg: 12, 90

- **Float and Double:** This is used to represent a number with decimal point.

Eg: 45.1, 67.3

- **Character :** This is used to represent single character

Eg: 'C', 'a'

- **String:** This is used to represent group of characters.

Eg: "C.M.R.I.T College"

- **Boolean:** This is used represent logical values either true or false.

39

Non-primitive data type: The data types that are derived from primary data types are known as non-Primitive data types. These data types are used to store group of values. The non-primitive data types are

- Arrays
- Structure
- Union
- linked list
- Stacks

Stack is an Abstract linear data structure (ADT) works on the principle Last In First Out (LIFO). ii.The last element added to the stack is the first element to be deleted. ii.Insertion and deletion can be takes place at one end called TOP. iv.It looks like one side closed tube.

Applications Of Stack:

1. Expression evaluation
2. Backtracking (game playing, finding paths, exhaustive searching)
3. Memory management, run-time environment for nested language features.
4. Parsing
5. Recursion
6. Expression conversions(Infix to postfix and Prefix, postfix and Prefix to Infix)
7. Towers Of Hanoi

Applicatio of Stack:- To find the Factorial of a number using Recursion

- Queue etc

5(a)	**Given two university information files "studentname.txt" and**	[10]	CO6	L3

"usn.txt" that contains students Name and USN respectively. Write a C program to create a new file called "output.txt" and copy the content of files "studentname.txt" and "usn.txt" into output file in the sequence shown below. Display the contents of output file "output.txt" on to the screen.

```
#include<stdio.h>
int main()
{
        FILE *fp1,*fp2,*fp3;
        char buff1[100],buff2[100];
        fp1=fopen("studentname.txt","r");
        fp2=fopen("usn.txt","r");
        fp3=fopen("output.txt","w");
        fprintf(fp3,"usn\tname\n");
                while(1)
                        {
                                fscanf(fp1,"%s",buff1);
                                fscanf(fp2,"%s",buff2);
                                if(!feof(fp1) && !feof(fp2))

        fprintf(fp3,"%s\t%s\n",buff1,buff2);
                                else
                                        break;
                        }
    fclose(fp1);
    fclose(fp2);
    fclose(fp3);
}
```

OR

(b) **Write note on a) Stack b) Queue c) Linked list d) Tree. (Mention Applications)** [10] | CO6 | L1

STACKS

• A stack is a special type of data structure where elements are inserted from one end and elements are deleted from the same end.

• Using this approach, the Last element Inserted is the First element to be deleted Out, and hence, stack is also called LIFO data structure.

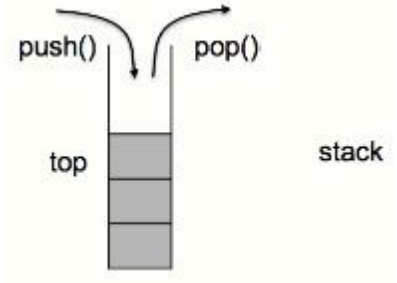
• The various operations performed on stack:

Insert: An element is inserted from top end. Insertion operation is called push operation.

Delete: An element is deleted from top end. Deletion operation is called pop operation.

Overflow: Check whether the stack is full or not.

Underflow: Check whether the stack is empty or not.



APPLICATIONS OF STACK

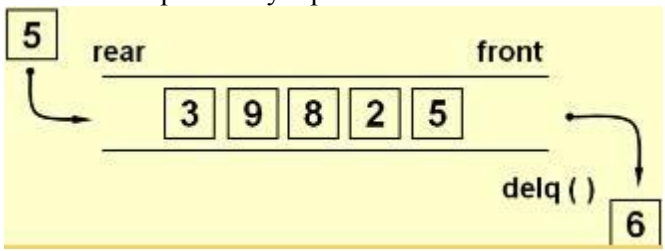
1) Conversion of expressions: The compiler converts the infix expressions into postfix expressions using stack.
2) Evaluation of expression: An arithmetic expression represented in the form of either postfix or prefix can be easily evaluated using stack.
3) Recursion: A function which calls itself is called recursive function.
4) Other applications: To find whether the string is a palindrome, to check whether a given expression is valid or not.

QUEUES

• A queue is a special type of data structure where elements are inserted from one end and elements are deleted from the other end.
• The end at which new elements are added is called the rear and the end from which elements are deleted is called the front.
• The first element inserted is the first element to be deleted out, and hence queue is also called FIFO data structure.
• The various operations performed on queue are

1) Insert: An element is inserted from rear end.
2) Delete: An element is deleted from front end.
3) Overflow: If queue is full and we try to insert an item, overflow condition occurs.
4) Underflow: If queue is empty and try to delete an item, underflow condition occurs.

• This can be pictorially represented as shown below:



LINKED LIST

• A linked list is a data structure which is collection of zero or more nodes where each node has some information.
• Normally, node consists of 2 fields

1) info field which is used to store the data or information to be manipulated
2) link field which contains address of the next node.

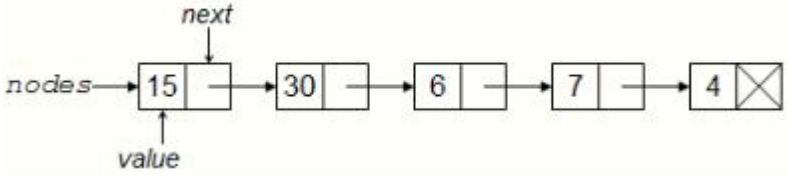
• Different types of linked list are SLL & DLL
• Various operations of linked lists

1) Inserting a node into the list
2) Deleting a node from the list
3) Search in a list
4) Display the contents of list

SINGLY LINKED LIST

• This is a collection of zero or more nodes where each node has two or more fields and only one link field which contains address of the next node.

• This can be pictorially represented as shown below:



BINARY TREE

• A tree in which each node has either zero, one or two subtrees is called a binary tree.

• figure

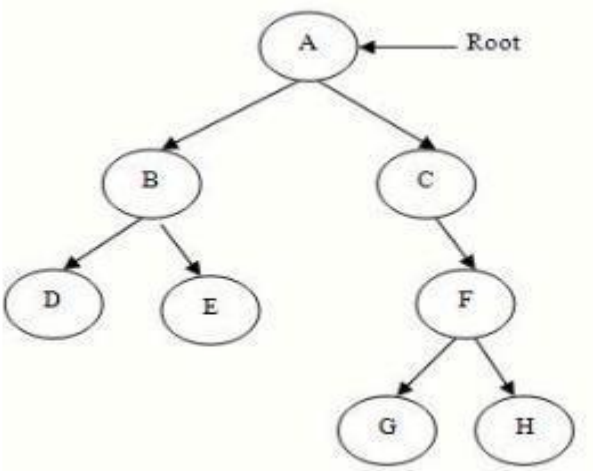
• Each node consists of three fields

llink: contains address of left subtree

info: this field is used to store the actual data or information to be manipulated

rrlink: contains address of right subtre

• This can be pictorially represented as shown below:



Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1:	Identify variable and their data types for a given problem	2	1	2	-	1	1	1	1	2	1	-	1
CO2:	Use operators to form a computational step.	2	1	2	-	1	1	1	1	2	1	1	1
CO3:	Use Control statement to solve simple algorithms - sort, search.	2	1	2	-	1	1	1	1	2	1	-	1
CO4:	Write functions that solve a given problem.	2	1	2	-	1	1	1	1	2	1	-	1
CO5:	Explain dynamic memory allocation using an example - Array of strings	2	1	2	-	1	1	1	1	2	1	-	1
CO6:	Explain basic data structures used for Programming - Arrays, List, Stack, Queue, Trees.	2	1	2	-	1	1	1	1	2	1	1	1

Cognitive level	KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PO1 - *Engineering knowledge*; PO2 - *Problem analysis*; PO3 - *Design/development of solutions*; PO4 - *Conduct investigations of complex problems*; PO5 - *Modern tool usage*; PO6 - *The Engineer and society*; PO7- *Environment and sustainability*; PO8 – *Ethics*; PO9 - *Individual and team work*; PO10 - *Communication*; PO11 - *Project management and finance*; PO12 - *Life-long learning*