CMR INSTITUTE OF TECHNOLOGY

Internal Assessment Test 1 – September 2016-SCHEME & SOLUTION

Sub:	Programming in C and Data Structures							**Code:**	15PCD13
Date:	07/ 09/2016	Duration:	90mins	Max Marks:	50	**Sem:**	I	**Branch:**	

Note: Answer any five questions:

1	**Define a variable. What are the rules for constructing variable in C programming language? How is a variable different from a constant? Justify your answer with a relevant programming example.**	10M

- A variable is a data name that may be used to store a data value.**(1M)**
- Variable names may consist of letters, digits, and the underscore(_) character, subject to the rules given below: **(3M)**
  1. The variables must always begin with a letter. Some systems permit underscore as the first character.

  2. ANSI standard recognizes a length of 31 characters. However, the length should not be normally more than eight characters.

  3. Uppercase and lowercase are significant. That is ,the variable Rate is not the same as rate or TOTAL.

  4. The variable name should not be a keyword.

  5. White space is not allowed.
- Variables may take different values at different times during execution, constants remain unchanged during the execution.**(1M)**
- #include<stdio.h> **(5M)**

```
#include<stdio.h>
int main()
{
inta,b,c=10,res; // a,b are varaiables& c is a constant with the value 10
printf("\n Enter the values of a & b:");
scanf("%d%d",&a,&b);
    res=a+b+c;
printf("The sum is %d",res);
    return 0;
}
```

2 **a) With a neat diagram, explain the structure of a C program. (5)** 10M

- The basic structure of a C program is shown below Documentation Section

Documentation Section
Link Section
Definition Section
Global Declaration Section
main() Function Section
{
Declaration Part Executable Part
}
Subprogram section
Function 1
Function 2
.
.
.
Function n

- The documentation section consists of a set of comment lines giving the name of the program, the name of the author and other details which the programmer would like to use later.
- The link sectionprovidesinstructions tothecompilerto linkfunctionsfromthesystemlibrary.
- The definition section contains all symbolic constants.
- There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declarationsection.
- Every C program must have one main() function section.
- The subprogram section contains all the user-defined functions that are called in the main function. The main function is very important compared to other sections.

b) Write a C program to convert an angle entered in degrees into radians. (5)
Formula: radians= (3.14/180)*degree

```
#include<stdio.h>
#include<math.h>
int main()
{
int i;
     float degree, x;
// Input the value of x in degree.
printf ( "\nEnter the value of x in degree: " );
scanf ( "%f", &degree );
// Compute the value of x in radians.
x = (3.1412/180.0) * degree;
printf("\n The value of angle in radians is %f",x);
   return 0;
}
```

3 **Define a token. Describe various token types in C programming language.** 10M

- A C program consists of various tokens and a token is either a keyword, an identifier, a constant, a string literal, or a symbol.**(2M)**
-

4 **What do you mean by two way selection statements? Explain if, if-else, nested if-else and cascaded if-else with an example.**

Two-Way Selection**(2M)**

A two way selection statement checks for a condition and selects from the two statement blocks which one to execute depending upon whether the condition is true or false.

Control statements are used to control the order of execution of the statements. Decision statements controls the execution/not execution of certain statements based on the condition. The main decision statement in C is if statement. There are four variants:

- if
- if-else
- nested if-else
- cascaded if-else or else-if ladder

If Statement (2M)

It is one-way selection statement. It is used only when there is one alternative. The syntax of if statement is:

```
if (expression )
{
        statement-block-1;
}
statement-block-2;
```

The expression is evaluated to true or false.

- If the expression is evaluated to true, then statement-block-1 is executed and the control comes outside of if statement and the execution of further statements (statement-block-2) continues if any.
- If the expression is evaluated to be false, then statement-block 1 is skipped.

Example: Any program to show the working of if-statement

If-else Statement (2M)

It is two-way selection statement. It is used when we have to choose between two alternatives.

The syntax of if-else statement is:

```
if(expression)
{
statement-block-1;
}
else
{
statement-block-2;
}
statement-block-3;
```

The expression is evaluated to true or false.

If the expression is evaluated to true, then statement-block-1 is executedand the control comes outside of if-else and statement-block-3 is executed.

If the expression is evaluated to false, then statement-block-2 is executedand the control comes outside of if-else and statement-block-3 is executed.

Example: Any program to show the working of if-statement

Nested if-else Statement (2M)

It is multi-way selection statement. It is used when an action has to be performed based on many decisions.

An if-else statement within another if-else statement: is called nested if-else statement.

The syntax of nested if-else statement is:

```
if ( expression-1 )
{
        if ( expression-2  )
                statement-block-1;
        else
                statement-block-2;

else
{
        if ( expression-3  )
                statement-block-3;
        else
                statement-block-4;
}
statement-block-5;
```

The expression-1 is evaluated to true or false.

If expression-1 is evaluated to true, then expression2 is evaluated to true or false. If expression-2 is evaluated to true, then statement-block-1 is executed. If expression-2 is evaluated to false, then statement-block-2 is executed.

If expression-1 is evaluated to false, then expression-3 is evaluated to true or false. If expression-3 is evaluated to true, then statement-block-3 is executed. If expression-3 is evaluated to false, then statement-block-4 is executed.

In either of the cases, after execution of a particular statement-block the control comes outside the nested if-else statement and continues execution from statement-block-5.

Example: Any program to show the working of Nested if-else statement

Cascaded if-else or else if ladder Statement (2M)
It is multi-way selection statement. It is used when we must alternatives.

The syntax of cascade if-else or else if ladder statement is:

```
if    (    expression-1    )
{
        statement-block-1;
.r
else if     expression-2  )
{
        statement-block-2;
}
else if     expression-3
{
        statement-block-3;
}
else if     expression-4
{
        statement-block-4;
}
else
{
        statement-block-5;

}
statement-block-6;
```

The expression is evaluated in top to bottom order. If an expressionis evaluated to true, then the statement-block associated with that expression is executed and the control comes out of the entire else if ladder and continues execution from statement-block-6 if any.

For example:

If expression-1 is evaluated to true, then statement-block-1 is executed and the control comes out of the entire else if ladder and continues execution from statement-block-6 if any.

If all the expressions are evaluated to false, then the last statement-block-5 (default) is executed and the control comes out of the entire else if ladder and continues execution from statement-block-6 if any.

Example: Any program to show the working of Cascaded if-else statement

5 **a) What are loop control statements? With proper syntax, explain each of them with a sample program.**
b) Differentiate between while, do-while and for loops based on the scenario of its usage. 10M

a) A set of statements that are executed repeatedlyuntil a conditionissatisfiediscalled a loop. When a condition is not satisfied then the loop is terminated.It usefulinperforming repetitive tasks. There are 3 types of loops:

- while
- d0-while
- for

while Loop (2M)

A while loop is used to execute a set of statements repeatedly as long as the given condition is satisfied.

The syntax of while loop is:

```
while ( expression )
{
        statement-block-1;
}
statement-block-2;
```

The expression is evaluated **to true or false.**

■ If the expression is evaluated to true (1), then the body of the loop (statement-block-1) is executed. After executing the body of the loop, the control goes back to the beginning of the while statement and the expression is evaluated to true or false. This cycle is continued until the expression becomes false.

- If the expression is evaluated to false (0), then the control comes out of the loop, without executing the body of the loop and the starts executing from statement-block-2.

Example: Any program to show the working of While loop.

do-while Loop (2M)

A do-while loop works similar to a while loop, except execute the body of the loop at least one time.

The syntax of do-while loop is:

```
        do
          {
            Statement-block-1;
          }
     while (  expression );
           statement-block-2;
```

The body of the loop (statement-block-1) is executed expression is evaluated to true or false.

■ If the expression is evaluated to true, then that it is guaranteed to execute at least once. Then the body of the loop (statement-block-1) is executed. After executing the body of the loop the expression is evaluated again to true or false. This cycle continues until the expression becomes false.

- If the expression is evaluated to false, then the control comes out of the loop and starts executing from statement-block-2.

Example: Any program to show the working of do- While loop.

For Loop (2M)

	A for loop is used to execute a set of statements repeatedly as long as the given condition is true.
	The syntax of while loop is:
	for (expression-1; expression-2; expression-3)
	{
	statement-block-1;
	}
	
	statement-block-2;
	Here, expression-1 contains initialization statement.
	expression-2 contains limit test expression.
	expression-3 contains updating statement.
	Expression-1 is evaluated first. It is executed only once i.e. when for loop is entered for the first time.
	Then expression-2 is evaluated to true or false.
	• If the expression-2 is evaluated to true, then the body of the loop (statement-block-1) is executed once. After executing the body of the loop, expression-3 is evaluated. Then the expression-2 is evaluated to true or false. If true the cycle continues until expression-2 becomes false.
	If the expression-2 is evaluated to false, then the control comes out of the loop, without executing the body of the loop and the starts executing from statement-block-2.
	
	Example: Any program to show the working of for loop.
	
	b) Diffrence between the 3 loops- (4M)
	While loop: used when you have an idea about the range of values on which to iterate, but don't know the exact number of iterations taking place.
	Do while loop: Similar to while loop except that the body of the loop will be executed at least once in the lifetime of the loop.
	For loop: used when you know the number of iterations beforehand.	

6	**Write a C program that reads three coefficients (a, b and c) of a quadratic equation of the form (ax2+bx+c) and compute all possible roots.**
	/*Program to find roots of a Quadratic Equation. */
	#include<stdio.h>
	#include<math.h>
	int main()
	{
	float a, b, c, d, rpart, ipart, r1, r2;
	// Input the co-efficients of a, b and c.
	printf ("\nEnter three non-zero coefficients (a, b and c) of the Quadratic Equation: ");
	scanf ("%f%f%f", &a, &b, &c);
	// Check if the equation is quadratic or not.
	if (a == 0)
	{
	printf ("\nThe equation is linear and not quadratic!!!\n");
	return 1;
	}
		10M

```
// Compute the discriminant.
d = ( b * b ) - ( 4 * a * c) ;
printf ( "\nThe discriminant is: %f\n", d );
// Compute Real and Equal roots.
if ( d == 0 )
{
r1 = r2 = -b / ( 2.0 * a );
printf ( "\nRoots are Real and Equal: \n r1= %f \n r2= %f \n\n", r1, r2 );
}
// Compute Real and Distinct roots.
else if ( d > 0 )
{
r1 = ( -b - ( sqrt ( d ) ) ) / ( 2.0 * a );
r2 = ( -b + ( sqrt ( d ) ) ) / ( 2.0 * a );
printf ( "\nRoots are Real and Distinct: \n r1= %f \n r2= %f \n\n", r1, r2 );
}
// Compute Imaginary roots.
else
{
rpart = - b / ( 2.0 * a );
ipart = sqrt ( ( - d ) ) / ( 2.0 * a );
printf ( "\nRoots are Imaginary: \n r1 = %f + i * %f \n r2 = %f - i * %f \n\n", rpart, ipart, rpart, ipart );
}
return 0;
}
```

7 **Write a C program to read a positive number, find its reverse and check whether it's a palindrome or not. Show the tracing to justify the correctness of your program. (Example: if the number is 2014 and reverse is 4102. Hence, it's not a palindrome).** 10M

/*Program to check whether the given number is PALINDROME or NOT. */ (8M)

```
#include<stdio.h>
int main()
{
intnum, temp, rev, rem;
// Input the number.
printf ( "\nEnter an integer: " );
scanf ( "%d", &num );
// Initialize.
temp = num;
rev = 0;
// Compute reverse of the entered number.
while ( temp != 0 )
{
rem = temp % 10;
rev = rev * 10 + rem;
temp = temp / 10;
}
// Display reverse number.
printf ( "\nThe reversed number is: %d\n", rev );
// Check if the entered number and reversed number are equal.
if ( rev == num )
printf ( "\nThe entered number %d is a PALINDROME\n\n", num );
else
printf (" \nThe entered number %d is NOT a PALINDROME\n\n", num );
return 0;
```

}

<u>Tracing (2M)</u>

```
Let num=2014
temp=2014
Loop 1:
    rem=4
    rev=4
    temp=201
Loop 2:
    rem=1
    rev=41
    temp=20
Loop 3:
    rem=0
    rev=410
    temp=2
Loop 4:
    rem=2
    rev=4102
    temp=0
```

8 Evaluate the following expressions. Justify your answer with step by step evaluation:-

a) int a=2, b=3; a <= b*a % b ; (1M)

Output:

a<=6%3

a<=0

<u>2 less than 0 is false:</u> The output will be 0

b) int a=2 , b=3, c; (2M)

```
c = ++a;
printf("%d\n",c);
c = ++b;
printf("%d\n",c);
c = (--a) + (b--);
printf("%d\n",c);
```

Output:

```
3
  4
  6
```

c) a=5 , b=6 i) a|b ii) a&b iii) a>>1 iv) b<<1

Output i:

a=5--→ 0 1 0 1

b=6--→ 0 1 1 0

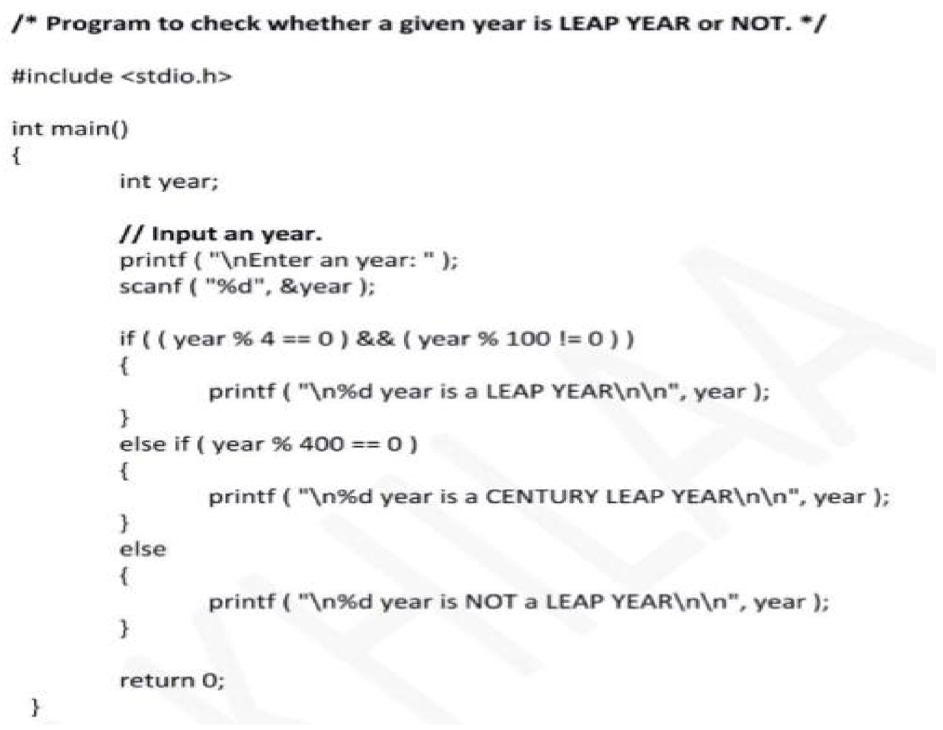
```
------------------
a|b--→ 0 1 1 1=7

Output ii:
a=5--→ 0 1 0 1
b=6--→ 0 1 1 0
------------------
a&b--→ 0 1 0 0=4

Output iii:
a=5--→ 0 1 0 1
------------------
a>>1--→ 0 0 1 0=2

Output iv:
b=6--→ 0 1 1 0
------------------
b<<1--→ 1 1 0 0=12
```

d) Write a C program to check whether the given year is a leap year or not. Also consider century years. (5)



```c
/* Program to check whether a given year is LEAP YEAR or NOT. */

#include <stdio.h>

int main()
{
            int year;

            // Input an year.
            printf ( "\nEnter an year: " );
            scanf ( "%d", &year );

            if ( ( year % 4 == 0 ) && ( year % 100 != 0 ) )
            {
                        printf ( "\n%d year is a LEAP YEAR\n\n", year );
            }
            else if ( year % 400 == 0 )
            {
                        printf ( "\n%d year is a CENTURY LEAP YEAR\n\n", year );
            }
            else
            {
                        printf ( "\n%d year is NOT a LEAP YEAR\n\n", year );
            }

            return 0;
  }
```