MODEL ANSWER

WINTER - 2017 EXAMINATION

Subject: Programming in 'C' **Subject Code: 17212**

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q. No.	Sub Q.N.	Answer	Marking Scheme
1.	 **(a)** **Ans.**	**Attempt any TEN of the following:** **Define global variable.** A global variable is a variable that is declared outside all functions. It can be used in all functions.	**20** **2M** ***Definition 2M***
	(b) **Ans.**	**State any four data types in 'C'.** ***(Note: Any other correct data type shall be considered)*** **Data types:** • int • float • double • char • void	**2M** ***Any four $^{1/2}$M each***
	(c) **Ans.**	**State any four string handling function.** ***(Note: Any other correct string function shall be considered)***	**2M**

<table>
<tr><td></td><td></td><td>1. strlen(stringvalue)
2. strcpy(string dest, string src)
3. strcat(string dest, string src)
4. strcmp(string str1, string str2)</td><td><i>Any four $^{1/2}$M each</i></td></tr>
<tr><td></td><td><b>(d)
Ans.</b></td><td><b>Define recursive function.</b>
<b>Recursive function:</b>
Recursion is the process of function calling itself again and again.</td><td><b>2M</b>
<i><b>Definition 2M</b></i></td></tr>
<tr><td></td><td><b>(e)
Ans.</b></td><td><b>State any four relational and logical operators.</b>
<b>Relational operators:</b>
<table>
<tr><th>Operator</th><th>Meaning</th></tr>
<tr><td>&lt;</td><td>Less than</td></tr>
<tr><td>&lt;=</td><td>Less than or equal to</td></tr>
<tr><td>&gt;</td><td>Greater than</td></tr>
<tr><td>&gt;=</td><td>Greater than or equal to</td></tr>
<tr><td>= =</td><td>Equal to</td></tr>
<tr><td>!=</td><td>Not equal to</td></tr>
</table>

<b>Logical operators:</b>
&&-logical AND
|| - logical OR
! – logical NOT</td><td><b>2M</b>
<i><b>Any four relational operators 1M</b></i>
<i><b>logical operators 1M</b></i></td></tr>
<tr><td></td><td><b>(f)
Ans.</b></td><td><b>State uses of * and & operators with respect to pointer.</b>
<b>*</b> - It is used to declare a pointer variable.
 int *ptr;
It is also used as value at operator.
printf(“%d”,*ptr);

<b>&amp;</b>-It is used to retrieve address from the memory.
int a,*ptr;
ptr=&amp;a;</td><td><b>2M</b>
<i><b>Use of each 1M</b></i></td></tr>
<tr><td></td><td><b>(g)
Ans.</b></td><td><b>State uses of continue statement.</b>
<b>Uses of continue statement:</b>
Continue statement is used to continue with the next iteration after skipping any statements in between.</td><td><b>2M</b>
<i><b>2M for use</b></i></td></tr>
<tr><td></td><td><b>(h)
Ans.</b></td><td><b>State any two advantages of function.</b></td><td><b>2M</b></td></tr>
</table>

		Advantages of function: • It facilitates top-down modular programming. • The length of a source program can be reduced by using functions at appropriate places. • It is easy to locate and isolate a faulty function. • A function may be used by many other programs i.e. a function written for one program can be used by other programs.	***Any two advantages 1M each***
	(i) **Ans.**	**State any two features of C language.** ***(Note: Any other relevant feature shall be considered)*** **Features of C language:** • It is a robust language with rich set of built-in functions and operators that can be used to write any complex program. • Programs written in C are efficient and fast. This is due to its variety of data type and powerful operators. • A C program is basically a collection of functions that are supported by C library. We can also create our own function and add it to C library. • C language is the most widely used language in operating systems and embedded system development today.	**2M** ***Any two features 1M each***

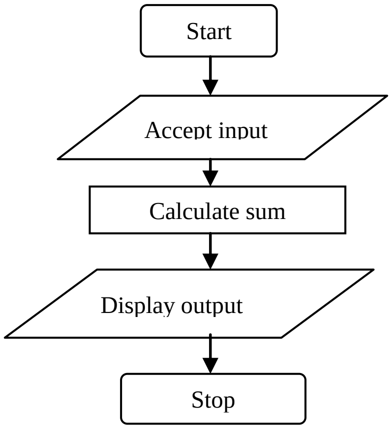
	(j)
	Ans.
		Write the Syntax of switch case statement.
		switch(variable/expression)
		{
		case value1:
		statements;
		break;
		case value2:
		statements;
		break;
		default:
		statements;
		break;
		}
			2M
			Correct syntax 2M

	(k) **Ans.**	**Define array. How one dimensional array is declared?** **Array:** An array is a collection of data elements of same data type. The values in an array are stored in continuous memory locations.	**2M** ***Definition 1M***

MODEL ANSWER

WINTER - 2017 EXAMINATION

Subject: Programming in 'C' **Subject Code: 17212**

		To declare a one dimensional array: datatype arr_name[size]; int arr[5] ;	***Declaration 1M***
	(l) **Ans.**	**Write output of following program:** **void main ( )** **{** **int m;** **for (m=1; m<5; m++)** **printf("%d \n", (m%2)? m: m * 2);** **}** **Output:** 1 4 3 8	**2M** ***Correct output 2M***
2.	 **(a)** **Ans.**	**Attempt any FOUR of the following:** **Explain the term flowchart and algorithm with an example.** **Flowchart:** A flowchart is a **diagrammatical** representation of solution for the given task. ***Example:***  Start Accept input Calculate sum Display output Stop **Algorithm**: An algorithm is a collection of statements to be performed in a sequence to solve a problem. ***Example:*** Step 1. Start	**16** **4M** ***1M*** ***Any example 1M*** ***1M***

```
Step 2. Accept the input for the two numbers: a and b
Step 3. Calculate sum  as a+b
Step 4. Display the output with sum
Step 5. Stop
```

Any example 1M

(b) Explain nested if-else with example. **4M**

Ans. **Nested if-else example:**

```
if(num<100)    ------------------Condition 1
  {
     if(num<50)   -----------------Condition 2
     {
       printf("Number is less than 50");   ----Statement 1
     }
     else
     {
       printf("Number is greater than 50 but less than 100");--- statement 2
      }
   }
   else
   {
         printf("Number is greater than 100");---statement 3
    }
```

Any correct Example 2M

Nested if else statement is used when multiple decisions are involved in a program.

With reference to above example:

If Condition 1 is true then condition 2 will be checked. If condition 2 is also true then control will pass to statement 1.

If condition 1 is true and condition 2 is false then control will pass to statement 2.

If condition 1 is false then control will directly pass to statement 3 skipping condition 2.

Explanation 2M

(c) State any four rules for choosing variable name. **4M**

Ans. **Rules for choosing variable name:**

1. Variable name must start with alphabet or underscore
2. Variable name may contain digits, alphabets in upper case or lower case or underscore.
3. No other special character, except underscore, is allowed in the variable name.
4. Blank spaces or white spaces are not allowed in the variable name

Any four rules 1M each

		5. Variable name should not be a reserved keyword.	

	(d)
	Ans.
		Write a program to display prime numbers between 1 to 50.
		#include<stdio.h>
		#include<conio.h>
		void main() {
		int num = 50, count,i,j;
		clrscr();
		printf("prime numbers are:");
		for (i = 1; i<= num; i++) {
		count = 0;
		for (j = 2; j <= i / 2; j++) {
		if (i % j == 0) {
		count++;
		break;
		}
		}
		if (count == 0) {
		printf("%d\n",i);
		}
		}
		getch();
		}
			4M
			Correct logic 2M
			Correct syntax 2M

	(e)
	Ans.
		Write a program to find transpose of 3 x 3 matrix.
		#include<stdio.h>
		#include<conio.h>
		void main(){
		int a[3][3],t[3][3];
		int i, j;
		clrscr();
		for(i=0;i<3;i++){
		for(j=0;j<3;j++){
		printf("Enter value");
		scanf("%d",&a[i][j]);
		}
		}
		printf("The elements of the matrix are:\n");
		for(i=0;i<3;i++){
		for(j=0;j<3;j++) {
		printf("%d\t",a[i][j]);
			4M
			Correct logic 2M
			Correct syntax 2M

		}
		printf("\n");
		}
		for(i=0;i<3;i++){
		for(j=0;j<3;j++) {
		t[j][i]=a[i][j];
		}
		}
		printf("The transpose matrix is: \n");
		for(i=0;i<3;i++){
		for(j=0;j<3;j++) {
		printf("%d\t",t[i][j]);
		}
		printf("\n");
		}
		getch();
		}	

	(f)
	Ans.
		Write a program to calculate factorial of number.
		#include<stdio.h>
		#include<conio.h>
		void main() {
		int fact = 1,n,i;
		clrscr();
		printf("Enter a number");
		scanf("%d",&n);
		for(i = 1; i <= n; i++) {
		fact = fact*i;
		}
		printf("%d",fact);
		getch();
		}
			4M
			Correct logic 2M
			Correct syntax 2M

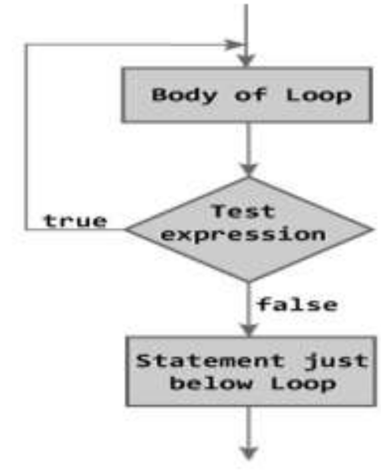
3.	 **(a)** **Ans.**	**Attempt any FOUR of the following:** **Explain static and external variables.** **1. static Variables :** • The value of Static Variables persists until the end of the program. • It may be Internal or External type depends on place of Declaration. • Internal Static variables are same as auto except they remain alive throughout the program.	**16** **4M** ***Explanation of static 2M***

		• Ex. static int x;
		static float y;
		
		2. external variables :
		• Variables that are both alive and active throughout the entire program are called as External variables.
		• External variables are also called as Global variables which can be accessed by any function in the program.
		• extern keyword is used to declare External keyword.
		• External variables are declared outside the function.
		Ex.
		int number;
		float length=7.5;
		main()
		{
		--
		--
		}
		Function1()
		{
		--
		--
		}
			Explanation of external 2M

	(b)
	
	Ans.
		Write a function to exchange values of two variables using call by reference.
		(Note: program with call by reference method shall be considered)
		
		Function call:
		swap(&x,&y);
		
		Function Definition:
		swap(int *x,int *y)
		{
		int temp;
		temp=*x;
		*x=*y;
		*y=temp;
		}
			4M
			
			Correct function with call by reference method 4M

	(c)
	
	Ans.
		Write a program to find length of given string without library function.
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int len,i=0;
		char str1[10];
		clrscr();
		printf("\n Enter string :");
		gets(str1);
		while(str1[i]!='\0')
		{
		i++;
		}
		printf("%d",i);
		getch();
		}
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

	(d)
	Ans.
		Explain break statement with an example.
		Break statement:
		• The break statement transfers the control out of loop such as for, while or a block of statement such as switch case.
		• When a break statement is encountered, it skips the remaining part of current iterations of the loop.
		• The break will exit only a single loop.
		• *Syntax* : break;
		
		Example:
		switch(choice)
		{
		case 1:
		printf("welcome to case 1");
		break;
		case 2:
		printf("welcome to case 2");
		}
		Break statement used with case 1 will exit switch statement when case 1 completes its execution.
		If break statement is not used then after execution of case 1 compiler
			4M
			
			Explanation 2M
			
			Any Example 2M

		will execute case 2.	
	(e) **Ans.**	**Explain the use of do-while statement. Also draw the flow-chart for the same.** **Do-While statement:** • In some applications it is necessary to execute the body of the loop before the condition is checked; such situation can be handled by do statement. • At least once the body of loop will be executed. • do statement, first executes the body of the loop. • At the end of the loop, the test condition in the while statement is evaluated. If the condition is true, then it continues to execute body of the loop once again. • This process continues as long as the condition is true. • When the condition becomes false, the loops will be terminated and the control goes to next statement after while statement. **Flowchart**: Do statement directly executes body of loop.  Body of Loop Test expression true false Statement just below Loop Figure: Flowchart of do...while Loop	**4M** ***Use 2M*** ***Flowchart 2M***

	(f)
	
	Ans.
		Write a program to reverse given integer number (input = 4567 reverse is 7654).
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int no,sum=0,rem;
			4M
			
			Correct logic 2M

		printf("Enter the number :");
		scanf("%d",&no);
		while(no>0)
		{
		rem=no%10;
		no=no/10;
		sum=sum*10+rem;
		}
		printf("\n sum=%d",sum);
		getch();
		}
			Correct syntax 2M

4.	(a)
	Ans.
		Attempt any FOUR of the following:
		Write a program that will obtain the length and width of rectangle from user and compute area and perimeter.
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		float len,wid,area,per,a;
		clrscr();
		printf("\n Enter length and Breadth of Rectangle:");
		scanf("%f%f",&len,&wid);
		area=len*wid;
		printf("\n Area of rectangle :%f",area);
		a=len+wid;
		per=2*a;
		printf("\n Perimeter of rectangle :%f",per);
		getch();
		}
			16
			4M
			Correct logic 2M
			Correct syntax 2M

	(b) Ans.	**State any four control statements.** ***(Note: Any four control statements shall be considered).*** **Control statements:** 1. If statement 2. If-else statement 3. Do Statement 5. Do-while statement 5. Switch statement 6. for statement 7. goto statement	**4M** ***Any four control statements 1M each***

8. break statement
9. continue statement

(c) **State any four difference between if and switch statement.** 4M

(Note: Any relevant difference shall be considered).

Ans.

if	**switch**
If statement uses multiple conditions for multiple choices.	Switch statement uses single expression/condition for multiple choices.
if statement evaluates integer, character, pointer or floating-point type or Boolean type.	Switch statement evaluates only character or integer value.
If the condition inside if statements is false, then by default the else statement is executed if written.	If the condition inside switch statements does not match with any of the cases, for that instance the default statement is executed if written.
If statement test for equality as well as for logical expression.	Switch statement test only for equality.

Syntax :
if(expression)
{
Statement1;
Statement2;
--
Statement n;
}
	Syntax :
	switch(expression)
	{
	case constant 1 : statements
	break;
	case constant 2 : statements
	break;
	-
	-
	case constant n: statements
	break;
	default : statements;
	}

Any four differences 1M each

(d) **Write a program to sort array element in descending order.** 4M

(Note: Any sorting logic shall be considered).

Ans.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i, j, temp,n,num[10];
clrscr();
```

Correct logic 2M

```
printf("\n Enter the size of an array :");
scanf("%d",&n);
printf("\n Enter the array elements :");
for(i=0;i<n;i++)
{
scanf("%d",&num[i]);
}
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(num[i]<num[j])
{
temp=num[i];
num[i]=num[j];
num[j]=temp;
}
}
}
printf("\n the numbers in Descending order are\n ");
for(i=0;i<n;i++)
{
printf("%d\n",num[i]);
}
getch();
}
```

Correct syntax 2M

(e) Ans. **Explain strcat and strcmp function with example.** **4M**

1. strcat() :

- This function joins two strings together

  Syntax : strcat(string1,string2);

  - String1 and string2 are character arrays
  - When this function is executed string2 is appended to string1 by removing null character at the end of string1.

1M

Example:

```
char str1[10],str2[5];
str1="abc";
str2="xyz";
strcat(str1,str2);
```

Any example 1M

```
printf("%s",str1);
```

2. **strcmp() :**
   This function compares two strings. If the strings are equal then function returns 0 and if they are not equal it returns some numeric value.
   Syntax :strcmp(string1,string2);

1M

Example:

```
char str1[10],str2[5];
str1=”abc”;
str2=”xyz”;
 if(strcmp(str1,str2)==0)
 {
 printf("strings are equal");
 }
 else
 {
 printf("strings are not equal");
 }
```

Any example 1M

(f) **Distinguish between global and local variable.** **4M**

(Note: Any other relevant point shall be considered.)

Ans.

Local Variable	Global Variable
Local variables are declared inside a function.	Global Variables are declared outside all function.
Local Variables cannot be accessed outside the function.	Global Variables can be accessed in any function.
Local Variables are alive only within a function.	Global Variables are alive till the end of the program.
Created when the function block is created and destroyed when it is deleted.	Remain in existence for the entire time a program is executing.

Any four points 1M each

5. **Attempt any FOUR of the following:** **16**

(a) **Write a program to print sum of odd numbers between 20 to 50.** **4M**

Ans.

```
#include<stdio.h>
#include<conio.h>
void main()
{
```

Correct logic 2M

		int i, sum=0;
		clrscr();
		for(i=20;i<=50;i++)
		{
		if(i%2!=0)
		sum=sum+i;
		}
		printf("sum of odd numbers from 20 to 50 : %d",sum);
		}
			Correct syntax 2M

	(b)
	Ans.
		Explain declaration and initialization of two dimensional array.
		Two dimensional array:
		The array which is used to represent and store data in a tabular form is called as two dimensional array. Such type of array is specially used to represent data in a matrix form.
		
		Declaration of two dimensional arrays:
		Syntax:-
		Data_type arrayname [row size] [column size];
		Eg :
		int arr[3][4];
		this will declare array “arr” with 3 rows and 4 columns.
		
		Initialization can be done as design time or runtime:
		1. Design time: This can be done by providing row X column= number of elements to the array.
		Eg for a 3 rows and 4 columns array , 3X4=12 elements can be provided as :arr[3][4]={ { 2,3,4,6},{1,4,6,3},{6,6,4,3}};
		
		2. **Runtime:** For this loop structures like for, can be used in a nested form, where outer loop will increment row and inner loop will increment column.
		Eg :
		for(i=0;i<3;i++)
		{
		for(j=0;j<4;j++)
		{
		scanf(“%d”,&arr[i][j]);
		}
		}
			4M
			
			Explanation of Declaration 2M
			
			Explanation of Initialization 2M

	(c) **Ans.**	**Explain structure with suitable example.** **Structure:** A structure is a collection of one or more variables of same or different data types grouped together under a single name. Struct structure_name { Data_type variable 1; Data_type variable 2; . . . Data_type variable n; }variable_name; Structure variable is used to access members of structure inside main function with dot operator. Variables of structure can be declared as: Variable of structure can be declared at the end of structure declaration before semi colon or inside the main function. struct book b; for a single book struct book b[5] to store data of 5 books. ***Example:*** struct book { char tit[20]; char auth[20]; int price; }b1;	**4M** ***Explanation of structure 2M*** ***Any Example 2M***
	(d) **Ans.**	**Explain recursive function with an example.** **Recursive function:** Recursion is the process of function calling itself again and again. A recursive function contains function call to itself in the body of function. ***Example:*** #include<stdio.h> #include<conio.h> int factorial(int n);	**4M** ***Explanation of recursive function 2M***

		void main()
		{
		int n,fact;
		clrscr();
		printf("enter the number");
		scanf("%d",&n);
		fact=factorial(n);
		printf("factorial of %d=%d",n,fact);
		getch();
		}
		int factorial(int n)
		{
		if(n==1)
		{
		return(1);
		}
		else
		{
		return(n * factorial(n-1)); -------------------------Recursive function call
		}
		}
		In the above example recursive function factorial() is used to print the Factorial of a number.
			Any Example 2M

	(e) **Ans.**	**Explain arithmetic operators that are used with pointer variable.** In C, pointer holds address of a value, so there can be arithmetic operations on the pointer variable. Following arithmetic operations are possible on pointer in C language: 1. Increment 2. Decrement 3. Addition 4. Subtraction **1. Increment Operator (++):** Increment operation depends on the data type of the pointer variable. If it is an integer pointer ++ will increment address by 2 locations as it requires 2 bytes of storage in c. Similarly for float it will add 4 as float type requires 4 bytes of storage. **2. Decrement operator (--) :** Decrement operation depends on the data type of the pointer variable. If	**4M** ***Any four operators 1M each***

		it is an integer pointer -- will decrement address by 2 locations as it requires 2 bytes of storage in c. Similarly for float it will subtract 4 as float type requires 4 bytes of storage. **3. Addition (+) :** Here also the pointer address will change according to the data type. For example if ptr is an integer pointer ptr+2 will increment the address by 2* storage size required for integer. That means ptr will show an increment of 4 locations. **4. Subtraction (-) :** The pointer decrements according to storage size. For example ptr-2 will decrement the address by 2 locations if it is an integer pointer because integer requires storage size of 2 bytes.	
	(f) **Ans.**	**Define pointer. Describe & (ampersand) and * (asterisk) operators in pointers.** **Pointer:** A pointer is a variable used to store the memory address of the variable of similar data type. Variables store the values and pointers stores their addresses at which these variables are located. The values stored in the pointers are unsigned integer values. **& and * operators :** *Syntax to declare a pointer :* Syntax:-data_type * name_of_variable; 1) In declaration statement of pointer, name of pointer variable is preceded by * (indirection operator) operator. Also value at address can be retrieved by * operator. 2) An address of a variable can be assigned to a pointer variable with the help of &. ***Example :*** **Integer pointer :** int *ptr; --------------declaration of pointer variable int a=5; ptr = &a; ------------------ assignment of address here *ptr returns 5 as value at address where 'a' is stored.	**4M** ***Definition of pointer 1M*** ***Explanation of & and * operators $1^{1/2}$ M each***

6. Attempt any FOUR of the following: **16**

(a) Explain any four bitwise operators used in 'C' with example. **4M**

Ans. Bitwise operators:

| – Bitwise OR
& – Bitwise AND
~ – Bitwise NOT
^ – Bitwise XOR
<< – left shift
>> – right shift

Any four bitwise operators 1M each

Explanation:

Bitwise OR – |

It takes 2 bit patterns, and performs OR operations on each pair of corresponding bits.

The following example will explain it.

```
1010
1100
--------
OR 1110
```

Bitwise AND – &

It takes 2 bit patterns, and perform AND operations with it.

```
1010
1100
------
AND 1000
-------
```

The Bitwise AND will take pair of bits from each position, and if only both the bit is 1, the result on that position will be 1. Bitwise AND is used to Turn-Off bits.

Bitwise NOT

One's complement operator (Bitwise NOT) is used to convert each "1-bit to 0-bit" and "0-bit to 1-bit", in the given binary pattern. It is a unary operator i.e. it takes only one operand.

```
1001
----
NOT 0110
-------
```

		Bitwise XOR ^ Bitwise XOR ^, takes 2 bit patterns and perform XOR operation with it. 0101 0110 ------ XOR 0011 ------ **Left shift Operator – <<** The left shift operator will shift the bits towards left for the given number of times. int a=2<<1 **Right shift Operator – >>** The right shift operator will shift the bits towards right for the given number of times. int a=8>>1;	

	(b)
	Ans.
		Write a function to find largest number in an integer array.
		#include<stdio.h>
		void main()
		{
		largest();
		}
		void largest()
		{
		int i, arr[5], max=0;
		printf("Enter 5 integers of an array :");
		for(i=0;i<5;i++)
		{
		scanf("%d", &arr[i]);
		}
		//to find largest
		for(i=0;i<5;i++)
		{
		if(max<=arr[i])
		max=arr[i];
		}
		printf("Elements from array:\n");
		for(i=0;i<5;i++)
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

		{
		printf("%d\n",arr[i]);
		}
		printf("largest number : %d",max);
		}	

	(c)
	
	Ans.
		Write a program to show use of array of pointers.
		(Note: Any other relevant program with array of pointers shall be considered)
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int *ptr[5],i;
		int arr[]={1,2,3,4,5};
		clrscr();
		for(i=0;i<5;i++)
		{
		ptr[i]=&arr[i];
		}
		//display elements using pointer to array
		printf("Elements of array are :\n");
		for(i=0;i<5;i++)
		printf("%d ",*ptr[i]);
		}
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

	(d) Ans.	**Explain declaration and initialization of pointer variable.** Pointer is variable used to store the memory address of the variable. Variables store the values and pointers stores their addresses at which these variables are located. The values stored in the pointers are integer values. **Pointer declaration &initialization:** In declaration statement of pointer name of pointer variable is preceded by * (indirection operator) operator. *Syntax:-* data_type * name_of_variable *Ex.* int *ptr ; Here ptr variable of data type integer pointer is declared as name of variable "ptr" is preceded by * (indirection operator) means that variable	**4M** ***Explanation of declaration 2M*** ***Explanation of initialization 2M***

ptr is a pointer variable.
Data type of pointer variable should similar to data type of variable of which it is going to store address.
Pointers are initialized by address of another variable& for accessing address of a variable "address of" operator is used. This operator will fetch address of variable to which it prefixes (proceeds).
ptr=&name_of_variable;

(e) **Write a program to declare a structure employee having name, designation and salary. Accept and display this information for five members.** **4M**

Ans.

```
#include<stdio.h>
#include<conio.h>
struct employee
{
char ename[20];
char desg[20];
int salary;
}s[5];
```

Declaration of structure 2M

```
void main()
{
int i;
clrscr();
//Accepting information
printf("Enter details of 5 employees:\n");
for(i=0;i<5;i++)
{
printf("Enter name :");
scanf("%s",s[i].ename);
printf("Enter designation :");
scanf("%s",s[i].desg);
printf("Enter salary :");
scanf("%d",&s[i].salary);
}
//displaying information
printf("The details of emplyoees are :\n");
for(i=0;i<5;i++)
{
printf("%s\t%s\t%d",s[i].ename,s[i].desg,s[i].salary);
printf("\n-------------------\n");
```

Main function 2M

		} }	
	(f)	**Write a program to display Floyd's triangle as follow:** **1** **2 3** **4 5 6** **7 8 9 10**	**4M**

	Ans.	#include<stdio.h>
		void main()
		{
		inti,j,k=1;
		clrscr();
		for(i=1;i<=4;i++)
		{
		for(j=1;j<=i;j++)
		{
		printf("%d ",k);
		k++;
		}
		printf("\n");
		}
		}
			Correct logic 2M
			Correct syntax 2M