MODEL ANSWER

SUMMER – 2018 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 following terms:** **(i) Variable (ii) Constant** **(i) Variable:** Variable is a symbolic name given to a memory location which holds some value inside it. **(ii) Constant:** A constant is a value inside an identifier which cannot be changed in the program.	**20** **2M** ***Each definition 1M***
	(b) **Ans.**	**Define the term loop.** **Loop:** In any programming language, a loop can be defined as a block in which certain number of statements can be sequentially executed in repetition until a condition becomes false.	**2M** ***Correct definition 2M***
	(c) **Ans.**	**State the use of strlen( ). Also give its syntax.** **Strlen( ):** strlen() is a string function which is used to find length of the string.	**2M** ***Use of strlen ( ) 1M***

MODEL ANSWER

SUMMER – 2018 EXAMINATION

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

		Syntax : strlen(string)	***Correct Syntax 1M***
	(d) Ans.	**Write any two advantages of using function.** **Advantages of using function:** 1) It can reduce length of the program. 2) Easy to locate errors and debug. 3) It implements top down approach. 4) It provides reusability i.e. once a function is written, it can be used at many places, even in other programs.	**2M** ***Any two advantages 1M each***
	(e) Ans.	**List two operators used with pointer.** ***(Note: Any other relevant operators shall be considered)*** **Operators used with pointers:** **&** - address of the variable ***** - value at the address	**2M** ***Any two operators 1M each***
	(f) Ans.	**State any four relational operators.** **Relational operators:** < Less than > Greater than <= Less than equal to >= Greater than equal to == Equal to != Not equal to	**2M** ***Any four operators - ½ M each***
	(g) Ans.	**Write syntax for scanf( ) function. Give one example.** **Syntax of scanf():** scanf("format specifier1 format specifier2,..format specifier n", &variable1, &variable2… &variable n); *Example:* Assuming a as an integer variable and b as a float variable, scanf() can be used to input values into a and b as : scanf("%d %f",&a,&b);	**2M** ***Syntax 1M*** ***Example 1M***
	(h) Ans.	**State the use of break statement.** **Use of break statement:** break statement is used to early exit from the loop/block. After exiting it transfers the control out of the loop/ block.	**2M** ***Correct use 2M***
	(i) Ans.	**Define the term character array.** Character array is the one which can be used to store sequence of characters inside it and can share a common name.	**2M** ***Definition 2M***

	(j)
	Ans.
		Write syntax to define function in 'C' program.
		Syntax:
		function_return_type <function name>([parameter list])
		{
		local variable declaration;
		executable statements;
		.
		.
		return[(variable)];
		}
			2M
			
			Correct Syntax 2M

	(k) **Ans.**	**Give output for following program:** **#include<stdio.h>** **void main( )** **{** **char ch = 'e';** **switch (ch)** **{** **case 'a':** **printf("I am in case a");** **case 'b':** **printf("I am in case b");** **default:** **printf("I am in default case");** **}** **Output:** I am in default case **OR** **Error** Code needs one curly brace bracket (}) at the end otherwise there will be an error showing missing curly brace bracket (}).	**2M** ***2M***

	(l)
	Ans.
		Write syntax of for loop.
		Syntax of for loop:
		for(initialization; condition; increment/decrement)
		{
		executable statements;
		.
		.
		}
			2M
			
			Correct syntax 2M

MODEL ANSWER

SUMMER – 2018 EXAMINATION

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

2.	**(a) Ans.**	**Attempt any FOUR of the following:** **Describe use of any two bit wise operators with suitable example.** **Bitwise operators:** | – Bitwise OR & – Bitwise AND ~ – One's complement ^ – Bitwise XOR << – left shift >> – right shift **Description:** **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 -------	**16** **4M** ***Description of any two bit wise operators 2M each***

		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 'C' program to calculate and display multiplication of 1 to 7 numbers using for loop.
		(E.g.: 1 * 2 * 3…. * 7 = 5040)
		(Note: Any other relevant logic shall be considered)
		#include<stdio.h>
		#include<conio.h>
		main()
		{
		int i,p=1;
		clrscr();
		for(i=1;i<=7;i++)
		{
		p=p*i;
		}
		printf("Multiplication of numbers from 1 to 7 is = %d",p);
		}
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

	(c) **Ans.**	**Describe following functions with its syntax and example:** **(i) strcmp( ) (ii) strcat( )** **(i) strcmp( ):** It is a string function, which is used to compare the contents of two strings. It returns 0 if both string are equal. Otherwise it returns the numerical difference between the ascii values of the first non matching pair of	**4M** ***Description with Syntax of strcmp***

		characters.
		Syntax:
		strcmp(string1,string2);
		
		Eg : if s1="there" and s2="their"
		the output of strcmp(s1,s2) will be 9 as the difference between ascii values of 'r' and 'i' is 9.
		
		(ii) strcat( ):
		It is string function which is to use concatenate second string at the end of the first string.
		Syntax:
		strcat(string1,string2);
		
		Eg: if s1="Msbte" and s2="Exam"
		strcat(s1,s2) will store value in s1 as "MsbteExam"
			( ) 1M
			
			Example 1M
			
			Description with Syntax of Strcat ( ) 1M
			
			Example 1M

	(d)
	
	Ans.
		Write a program to accept a number and display its cube using function.
		(Note: Any other relevant logic shall be considered)
		#include<stdio.h>
		#include<conio.h>
		void cube(int n)
		{
		printf("%d",n*n*n);
		}
		void main()
		{
		int no;
		clrscr();
		printf("\n Enter number:");
		scanf("%d",&no);
		cube(no);
		getch();
		}
			4M
			
			Correct logic 2M
			
			Correct Syntax 2M

	(e) **Ans.**	**Write a 'C' program to print length of accepted string using pointer.** ***(Note: Any other relevant logic shall be considered)*** #include<stdio.h> #include<conio.h>	**4M**

		void main()
		{
		char str[10],*ptr;
		int l=0;
		clrscr();
		printf("\n Enter string:");
		scanf("%s",&str);
		ptr=str;
		while(*ptr!='\0')
		{
		l=l+1;
		ptr=ptr+1;
		}
		printf("\n Length of string=%d",l);
		getch();
		}
			Correct logic 2M
			
			Correct Syntax 2M

	(f) Ans.	**Describe the use of continue statement with example.**
		Continue statement:
		Continue statement is used to continue the loop with the next iteration after skipping any statement in between. The continue statement tells the compiler that "skip the following statements and continue with the next iteration".
		Syntax:
		continue;
		Example:
		for (int j=0; j<=8; j++)
		{
		if (j==4)
		continue;
		printf("%d ", j);
		}
		In the above example, Value 4 is not displayed because when j=4 continue statement skips printf() statement and continues with next iteration of for.
		Output: 0 1 2 3 5 6 7 8
			4M
			
			Use of Continue Statement 2M
			
			Example 2M

3.	**(a)** **Ans.**	**Attempt any FOUR of the following:** **Describe with suitable example the concept of formatted output.** ***(Note: Example can be a program or a code snippet)*** **Formatted output:**	**16** **4M**

		printf( ) is used for formatted output to standard output depending on the format specification. Format specifiers, along with the data to be output are the parameters to the function. The different format specifiers used are: %d-int values %f-float values %c-char values %s-string *General syntax:* printf("control string/format specifier",data1,data2,…data n); control string indicates how many arguments follow and their data types. data1,data2 are the variables whose data are formatted and printed according to the specifications of the control string. *Example:* printf("%d %d",no1,no2); In the above example %d specify format and no1, no2 are the variables whose value will be display on the screen.	***Description 3M*** ***Example 1M***
	(b)	**Write a 'C' program to find gross salary of employee. Accept basic salary from user. If basic salary is less than 2000 then calculate HRA = 11% and DA = 80% of basic salary. If basic salary is equal or greater than 2000 then HRA = 600 and calculate DA = 95% of basic salary. Display gross salary. (gross salary = basic salary + HRA + DA)** ***(Note: Any other relevant logic shall be considered)***	**4M**

	Ans.	#include<stdio.h>
		#include<conio.h>
		void main()
		{
		float g_sal=0;
		float b_sal;
		float hra;
		float da;
		clrscr() ;
		printf("Enter the basic salary");
		scanf("%f",&b_sal);
		if(b_sal < 2000)
		{
			Correct syntax 2M
			
			Correct logic 2M

```
            hra=b_sal*0.11;
            da=b_sal*0.80;
            g_sal=b_sal+hra+da;
        }
        else if(b_sal >=2000)
        {
            hra=600;
            da=b_sal*0.95;
            g_sal=b_sal+hra+da;
        }
        else
        {
            printf("Not valid");
        }
        printf("Gross salary is %f",g_sal);
        getch();
}
```

(c) **Describe 'No argument with return value' category of function with example.** **4M**

Ans.

No argument with return value' category function does not accept any argument but returns a value as a result of function execution. This function is declared and defined with data type of return value and function name in a prototype. ***Description 2M***

Syntax:

```
Return type function_name( )
{
      Body of the function;
}
```

Example:

```
#include<stdio.h>
#include<conio.h>
int printNum();
void main()
{
   int i = printNum();
   printf("%d",i);
   getch();
}
```

Example 2M

		int printNum()
		{
		int i = 10;
		clrscr();
		return i;
		}	

	(d)
	
	Ans.
		Write a 'C' program to find factorial of a number using recursion.
		(Note: Any other relevant logic shall be considered)
		#include<stdio.h>
		#include<conio.h>
		int factorial(int num)
		{
		if(num==1)
		{
		return 1;
		}
		else
		{
		return(num*factorial(num-1));
		}
		}
		void main()
		{
		int num;
		int result;
		clrscr();
		printf("Enter a number");
		scanf("%d",&num);
		result=factorial(num);
		printf("Factorial of %d is %d",num,result);
		getch();
		}
			4M
			
			Correct syntax 2M
			
			Correct logic 2M

	(e) **Ans.**	**With suitable example, describe any two operations on pointer.** ***(Note: Code snippet shall be considered).*** The pointer arithmetic is done as per the data type of the pointer. The basic operations on pointers are **Increment:** It is used to increment the pointer. Each time a pointer is	**4M**

		incremented, it points to the next location. ***Example:*** For an int pointer variable, if the current position of pointer is 1000, when it is incremented, it points to 1002 because for storing an int value it takes 2 bytes of memory. int *ptr; ptr++; **Decrement:** It is used to decrement the pointer. Each time a pointer is decremented, it points to the previous location. ***Example:*** if the current position of pointer is 1002, then decrement operation results in the pointer pointing to the location 1000. int *ptr; ptr--; **Addition and subtraction** C Allows to add integers to or subtract integers from pointers as well as to subtract one pointer from another. **Example:** int*p1,*p2; p1+4; p2-2; p1-p2; **Comparison operators** Pointers may be compared by using relational operators, such as ==, <, and >. If p1 and p2 point to variables that are related to each other, such as elements of the same array, then p1 and p2 can be compared using the comparison operators.	***Any 2 operators description 1M each and example 1M each***
	(f) **Ans.**	**Write a 'C' program to accept a string from user and copy it into another string. Display both the strings.** ***(Note: Any other relevant logic shall be considered)*** #include<stdio.h> #include<conio.h> void main() { int i; char str[20]; char dest[20];	**4M** ***Correct syntax 2M***

```
    clrscr();
    printf("Enter a string");
    scanf("%s",str);
    for(i=0;str[i]!='\0';i++)
    {
          dest[i]=str[i];
    }
    dest[i]='\0';
    printf("The source string is %s",str);
    printf("\nThe copied string is %s",dest);
    getch();
}
```

OR

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char source[20];
    char dest[20];
    clrscr();
    printf("enter the string");
    scanf("%s",source);
    strcpy(dest,source);
    printf("Source string is %s",source);
    printf("destination string is %s",dest);
    getch();
}
```

Correct logic 2M

4.	**(a) Ans.**	**Attempt any FOUR of the following:** **Describe conditional operator with syntax and example.** **Conditional operator:** Conditional operators return one value if condition is true and returns another value if condition is false. This operator is also called as ternary operator as it takes three arguments. *Syntax*: (Condition? true_value: false_value); ***Example:*** #include<stdio.h> #include<conio.h> void main()	**16** **4M** ***Description with syntax 2M*** ***Example 2M***

		{
		int a,b,max;
		clrscr();
		printf("Enter the value of a & b:");
		scanf("%d%d",&a,&b);
		max=(a>b)?a:b;
		printf("%d is large",max);
		getch();
		}	

	(b)
	
	Ans.
		Describe with example in which case do-while loop is most suitable than while loop.
		Do-while is most suited when some statements of a program to be executed at least once even if the condition is false. When while loop is used, if the condition is false then the statements will not be executed even once.
		In menu driven programs do while loop is most suitable than while loop as do loop displays menu options at least once without checking any condition which is not possible with while loop.
		Example:-
		do
		{
		printf("Menu");
		printf("\n1. Add \n2.Subtract");
		printf("\n Enter your choice:");
		scanf("%d",&ch);
		switch(ch)
		{
		case 1:
		...
		...
		}
		printf("\n do you want to continue:(1/2)");
		scanf("%d",&ch);
		}while(ch==1);
		In the above example, menu will be displayed without checking any condition. Depending upon user's choice a case from switch will execute. If user wish to continue then while loop takes the control back to do statement.
			4M
			
			Description 1M
			
			Any suitable example with description 3M

		Displaying menu with while condition must have a condition to enter inside loop.	
	(c)	**Write a 'C' program to calculate and display sum of five elements from array.**	**4M**
		(Note: Any other relevant logic shall be considered)	

	Ans.	#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int arr[5];
		int i;
		int sum=0;
		clrscr();
		for(i=0;i<5;i++)
		{
		printf("Enter the nos of the array");
		scanf("%d",&arr[i]);
		}
		for(i=0;i<5;i++)
		{
		sum=sum+arr[i];
		}
		printf("Sum of the elements :%d",sum);
		getch();
		}
			Correct logic 2M
			Correct syntax 2M

	(d)	**Write a 'C' program to define a structure 'Bank' with members as branchno and bankname. Accept and display data for one bank.**	**4M**
		(Note: Any other relevant logic shall be considered)	

	Ans.	#include<stdio.h>
		#include<conio.h>
		void main()
		{
		struct bank
		{
		int branchno;
		char branchname[20];
		}b;
		clrscr();
			Correct syntax 2M
			Correct logic 2M

```
  printf("enter the branchno, branchname");
  scanf("%d%s",&b.branchno,&b.branchname);
  printf("The details of bank are\nNo: %d\nName:%s", b.branchno,
   b.branchname);
  getch();
 }
```

(e) **Differentiate between call by value and call by reference methods. (Any four points)** **4M**

(Note: Any other relevant points shall be considered)

Ans.

Call by value	Call by reference
In call by value, a copy of actual arguments is passed to respective formal arguments.	In call by reference, the location, that is, the address of actual arguments is passed to formal arguments
Actual arguments will remain safe, they cannot be modified in the called function.	Alteration to actual arguments is possible within called function.
Address of the actual and formal arguments are different	Address of the actual and formal arguments are the same
Changes made inside the function is not reflected in other functions	Changes made in the function is reflected outside also.

Any four points 1M each

(f) **Define the terms pointer and pointer expression. Also write two advantages of using pointer.** **4M**

Ans.

Pointer: Pointer is a variable that stores the address of another variable which is of similar data type.

Eg: int i=3;

int *ptr = &i; here the address of i is stored in the pointer variable ptr.

Definition of pointer 1M

Pointer expression:

When a pointer variable is used in side an expression then it is called a pointer expression.

Eg: int *ptr;

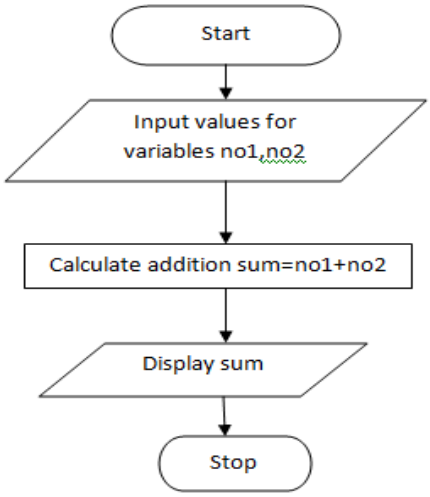
int i = 3;

ptr=&i;

ptr=ptr+3;

Definition of pointer expression 1M

.

		Advantages of using pointer: (i) It allows passing of arrays and strings to functions more efficiently. (ii) It makes possible to pass address of structure instead of entire structure to the functions. (iii) It makes possible to return more than one value from the function. (iv) It supports dynamic memory management.	***Any 2 advantages 1M each***
5.	**(a) Ans.**	**Attempt any FOUR of the following:** **Write an algorithm and draw a flowchart to add two numbers.** **Algorithm:** step 1: Start step 2: Input values for variables no1 and no2. step 3: Calculate addition of two values using formula as sum = no1 + no2. step 4: Display addition step 5: Stop **Flowchart:**  	**16** **4M** ***Correct algorithm 2M*** ***Correct flowchart 2M***
	(b) Ans.	**Describe importance of break statement in switch case statement.** **Break statement:** The break statement transfers the control out of loop/ block such as for, while or switch case. When a break statement is encountered, it	**4M**

		skips the remaining part of current iteration of the loop and passes control out of the block or loop. The break will exit only a single loop.
		Syntax:
		break;
		Break statement is important in switch case statement to allow only one case to execute from multiple cases. After executing statements from a single case, control must be pass outside the switch so that other cases cannot execute. If break statement is not given after any case then all cases after that case will also execute.
		Example:
		switch(choice)
		{
		case 1:
		printf("welcome to case 1");
		break;
		case 2:
		printf("welcome to case 2");
		case 3:
		printf("welcome to case 3");
		}
		In the above example, if choice is 1 then first case will execute and then break statement will transfer the control out of the block.
		If choice is 2 then first case will be ignored and second case executes. For second case break statement is not given so after executing second case, third case will also execute.
			Relevant description 4M

	(c)
	
	Ans.
		Write a 'C' program to read two matrices of 3 x 3. Calculate and display their addition.
		(Note: Any other relevant logic shall be considered)
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int a[3][3],b[3][3],c[3][3],i,j;
		clrscr();
		printf("Enter first matrix elements:\n");
		for(i=0;i<3;i++)
		{
		for(j=0;j<3;j++)
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

```
{
 scanf("%d",&a[i][j]);
 }
 }
 printf("\nEnter second matrix elements:\n");
 for(i=0;i<3;i++)
{
 for(j=0;j<3;j++)
{
scanf("%d",&b[i][j]);
}
}
 for(i=0;i<3;i++)
{
 for(j=0;j<3;j++)
{
 c[i][j]=a[i][j]+b[i][j];
}
}
printf("\n\nAddition of two matrices is:");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
printf("%d\t",c[i][j]);
}
}
 getch();
}
```

(d) **Write a 'C' program to implement a menu for following : 4M**
(i) To find whether the number is even or odd.
(ii) To find whether the number is positive or negative.
(Note: Any other relevant logic shall be considered)

Ans.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int choice,no;
clrscr();
```

		printf("\n Enter Number:");
		scanf("%d",&no);
		printf("\n Menu \n 1. Find whether number is even or odd");
		printf("\n 2. Find whether number is positive or negative");
		printf("\n Enter your choice:");
		scanf("%d",&choice);
		switch(choice)
		{
		case 1:
		if(no%2==0)
		printf("\nNumber is Even");
		else
		printf("\nNumber is Odd");
		break;
		case 2:
		if(no>0)
		printf("Numer is Positive");
		else
		printf("Number is Negative");
		}
		getch();
		}
			Correct logic 2M
			
			Correct syntax 2M

	(e) Ans.	**Describe register and static storage classes with example.**
		Register storage class: These variables are stored in the CPU registers instead of memory. Since the register access is much faster compared to the memory, frequently used variables can be stored this way. It is local to the block in which the variable is defined. It exists till the control remains within the block in which the variable is defined.
		Example:
		void main()
		{
		register int count=0;
		count++;
		printf("%d",count);
		}
		
		Static storage class: The value of the static variable persists until the end of the program execution. A variable can be declared as a static
			4M
			
			Description 1M each
			
			Example 1M each

		using the keyword static. The static variable is initialized to zero. It is local to the block in which the variable is defined. Value of the variable persists between different function calls.
		
		Example:
		void func1()
		{
		static int x=0;
		x= x+1;
		printf("x=%d",x);
		}	

	(f)	**State the meaning of each statement :** **int *ptr, no;** **no = 5;** **ptr = &no;** **printf("%d %d", no,*ptr);**	**4M**
	Ans.	**int *ptr,no;** -> This statement declares a pointer variable 'ptr' and a variable 'no' with data type as integer. **no=5;** -> This statement initializes value 5 to variable 'no' i.e. value 5 is stored in variable 'no' **ptr=&no;** -> This statement initializes pointer variable 'ptr' with the address of variable 'no' i.e it stores the address of variable 'no' in pointer variable ptr. **printf("%d%d",no,*ptr);** ->This statement displays value of no as 5 and value 5 stored in the address of 'no' within pointer variable.	***Meaning of each step 1M***
6.		**Attempt any FOUR of the following:**	**16**
	(a)	**Write a 'C' program to accept radius of circle and calculate area of circle. Display calculated area.** ***(Note: Any other relevant logic shall be considered)***	**4M**

	Ans.	#include<stdio.h>
		#include<conio.h>
		void main()
		{
		float radius,area;
		clrscr();
		printf("\n Enter radius:");
			Correct logic 2M
			
			Correct Syntax 2M

		scanf("%f",&radius);
		area=3.14*radius*radius;
		printf("\n Area =%f",area);
		getch();
		}	

	(b)
	
	Ans.
		Write a 'C' program to print sum of digits in the number.
		(e.g. number = 2134, sum = 2 + l + 3 + 4 = l0)
		(Note: Any other relevant logic shall be considered)
		#include<stdio.h>
		#include<conio.h>
		main()
		{
		int no,q,r,sum=0;
		clrscr();
		printf("enter a number:");
		scanf("%d",&no);
		while(no!=0)
		{
		r=no%10;
		sum=sum+r;
		q=no/10;
		no=q;
		}
		printf("sum of digits=%d",sum);
		getch();
		}
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

	(c) **Ans.**	**Define array. With suitable example, describe how to declare and initialize one dimensional array.** **Array:** An array is a collection of data elements of similar data type. The values in an array are stored in continuous memory locations. **Declare and initialize one dimensional array:** **syntax:-** datatype arr_name[size] ={values}; In the above syntax data type specify type of all data element stored inside an array. arr_name specify name of the array variable. Size specify number of data elements that can be stored inside it. Values are the data elements stored in an array.	**4M** ***Definition 2M*** ***Declaration 1M*** ***Initialization 1M***

		Example:- int a[10] = {10, 20, 5, 3, 55, 45, 15, 7, 30, 52}; In the above example, an array variable a stores 10 integer values inside it.	

	(d)
	
	Ans.
		Describe how to access and initialize structure members with example.
		Accessing member:
		Accessing structure members:-
		Structure members are accessed with structure variable and dot operator.
		Syntax:- structure_variable.structure_member
		Example:-
		struct student
		{ int **rollno**;
		char **nam**e[10];
		}**S1**;
		void main()
		{
		printf("%d",**S1.rollno**);
		printf("%s",**S1.name**);
		}
		In the above example, structure student has two members as rollno and name.'S1' is a structure variable. To access members of structure student,'S1' variable is used inside main().Variable 'S1' followed by dot operator and member name is used to access members of structure.
		
		Initialization of structure members:-
		Structure members can be initialized while creating structure variable. All the values as per number of members are specified inside a curly bracket along with comma as a separator.
		
		Example:-
		struct student S1={1,"abc"};
		
		In the above example, S1 is a structure variable. Value 1 is initialized to rollno and "abc" is initialized to name stored inside structure variable S1.
			4M
			
			Description of access to structure member 2M
			
			Description of Initialization of structure member 2M

	(e) Ans.	**Explain local and global variable with example.**
		Local variable is a variable that is declared inside a specific function. It is available and used only inside the function. Only the function in which it is declared can access it. It exists until end of function or block in which it is declared.
		void function1()
		{
		int number1;
		}
		void function2()
		{
		int number2;
		}
		In above example, variable number1 is accessed only inside function1 not in function2.
		
		Global variable is a variable that is declared outside all functions. It is available and used throughout the entire program. It can be accessed by any function in the program. It exists as long as the program's execution doesn't come to an end.
		Example:
		int number;
		void main()
		{
		}
		void function1()
		{
		}
		In the above example 'number' is a global variable.
			4M
			Local variable 2M
			Global variable 2M

	(f) Ans.	**Describe use of if-else statement with syntax and example.**
		Use:
		If-else statement is a decision making statement and is used to control the flow of execution of statements. It allows the computer to evaluate the expression first and then depending on whether the value of the expression is true or false, it transfers the control to the particular statement block.
		Syntax of if-else statement:
		if (test expression)
		{
			4M
			Description 2M
			Syntax 1M

		True-block statement (s)
		}
		else
		{
		False-block statement (s)
		}
		Statement-x;
		Example:-
		int no;
		no=2;
		if(no>0)
		{
		printf("Number is positive");
		}
		else
		{
		printf("Number is negative");
		}
			Example 1M