MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)

MODEL ANSWER

SUMMER - 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 moreImportance (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 token? List tokens in 'C'.** **Token :** Any basic element recognized by c compiler is called as token. **List of tokens in 'C'** 1. Identifiers 2. Literals 3. Variables 4. Constant 5. Operators 6. Keywords	**20** **2M** ***Definition 1M*** ***List of token 1M***
	(b) Ans.	**List any four relational operators.** **Four relational operators:** < : Less than > : Greater than <= : Less than equal to >= : Greater than equal to == : Equal to	**2M** ***Any four relational operators ½M***

!=	Not equal to

each

(c) Find error in following program and justify it. **2M**

```
#include <stdio.h>
void Main( )
{
  int I, a[5] = {7, 5, 2, 1, 9, 14};
  printf("%f", a[2]);
  getch( );
}
```

Ans. **Errors:**

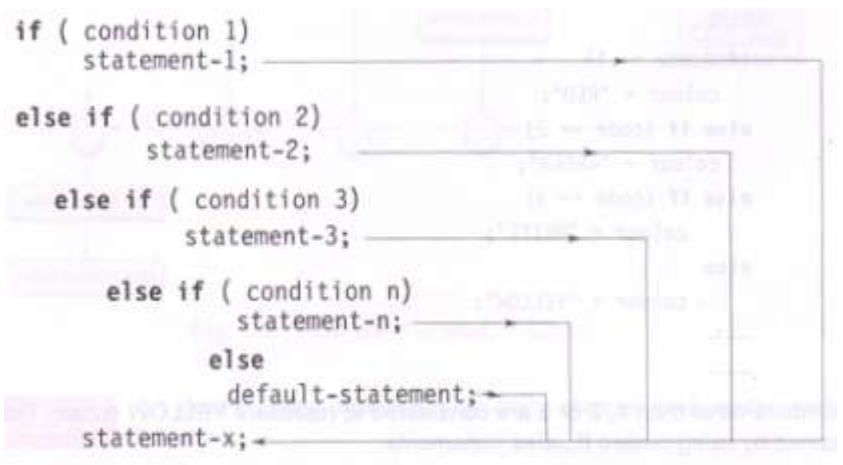
1) M should be small case in main()
2) Array size is 5 and elements are 6
3) %d should be there in place of %f
4) For getch() function there shall be conio.h Header file.

Any Two errors 1M each

(d) Write the syntax of else if-ladder. **2M**

Ans. **Syntax of else if…ladder:**

Syntax :



```
if ( condition 1)
    statement-1;
else if ( condition 2)
        statement-2;
  else if ( condition 3)
          statement-3;
     else if ( condition n)
             statement-n;
          else
            default-statement;
statement-x;
```

Correct syntax 2M

(e) State two differences between while loop and do while loop. **2M**

Ans.

While	**Do..while**
Entry controlled loop	Exit controlled loop
Condition is checked first	Condition is checked last
Executes only if satisfies the condition	Executes at least once even if the condition is not satisfied.
Syntax : while(condition)	Syntax: do

Any two differences 1M each

		{ Code; } | { Code; } while(condition);	
	(f) Ans.	**State the use and syntax of strcpy( ) function.** **strcpy() function:** **use :** It is a built in string library function which is used to copy one string to another. *Syntax:* strcpy(str1,str2); it copies contents of str2 to str1.	2M ***Use 1M*** ***Syntax 1M***

	(g) Ans.	**Give the syntax of switch case statement.**
		Syntax of switch case statement:
		switch(expression)
		{
		  case value1:
		      {
		        Statement;
		        break;
		      }
		case value2:
		      {
		        Statement;
		        break;
		      }
		.
		.
		.
		Default:
		     {
		      Statement;
		     }
		}
			2M
			
			Correct syntax 2M

	(h) Ans.	**State any two types of function on the basis of parameter passing and return type.** **Types of functions:** 1) Function with no return value and with parameter 2) Function with return value and with no parameter 3) Function with return value and with parameter	2M ***Any two types 1M***

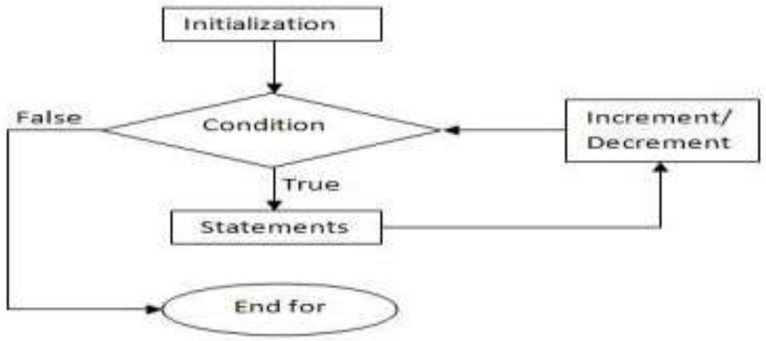
		4) Function with no return value and with no parameter	***each***
	(i) Ans.	**State the need of function.** **Need of a function:** 1. It provides modularity to the program. 2. Easy code Reusability. You just have to call the function by its name to use it. 3. In case of large programs with thousands of code lines, debugging and editing becomes easier if you use functions.	**2M** ***Any two points 2M***
	(j) Ans.	**Define structure.** Structure is a user-defined data type in C which allows you to combine different data types to store a particular type of record.	**2M** ***Definition 2M***
	(k) Ans.	**Define pointer.** **Definition**: A pointer is a variable that stores memory address of another variable which is of similar data type.	**2M** ***Correct definition 2M***
	(l) Ans.	**Write the use of indirection operator (*).** Indirection operator (*) is an operator used to obtain the value of a variable to which a pointer points.	**2M** ***Correct use 2M***
2.	**(a)** Ans.	**Attempt any FOUR of the following:** **Distinguish between variable and constant.**	**16** **4M** ***Any 4 points of difference 1M each***

Variable	**Constant**
A variable is a named location whose value can be changed during exe cution	A constant is a named location whose value cannot be changed during execution.
Syntax : datatype var_name;	Data type var_name=value; Or const data type var_name=value;
Eg : int a;	E.g: const int a= 5;
Variables can be initialized after declaration.	Constant have to be initialized at the time of declaration.
Dynamic declaration possible	Dynamic declaration not possible.

	(b) Ans.	**Write a program to find the sum of digit of an integer.** **(sum = 1 + 4 + 5 + 3 + 2 = 15)**	**4M**

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

		{
		int n,q,r,sum=0,n1;
		printf("enter a number:");
		scanf("%d",&n);
		n1=n;
		while(n!=0)
		{
		r=n%10;
		sum=sum+r;
		q=n/10;
		n=q;
		}
		printf("sum of digits of %d =%d",n1,sum);
		}
			Correct logic 2M
			
			Correct Syntax 2M

	(c)
	Ans.
		Describe use of for loop with syntax and flowchart.
		For loop can be used to execute some executable statements repeatedly. We can control iterations with an index variable. Initialization, condition to stop the loop and increment or decrement in index variable can be done in single statement in for loop.
		Syntax:
		for(initialization;condition;increment/decrement)
		{
		Code;
		}
		
		Flowchart of for loop:
		 Initialization
		False
		Condition
		Increment/ Decrement
		True
		Statements
		End for
			4M
			
			Use 1M
			
			Syntax 2M
			
			flowchart 1M

	(d) Ans.	**Write a program 'C' to find largest element from an array.** #include<stdio.h>	**4M**

		main()
		{
		int arr[5],max=0,i;
		printf("enter 5 elememts :");
		for(i=0;i<5;i++)
		{
		scanf("%d",&arr[i]);
		}
		max=arr[0];
		for(i=0;i<5;i++)
		{
		if(max<=arr[i])
		max=arr[i];
		}
		printf("Largest number = %d",max);
		}
			Correct logic 2M
			
			Correct Syntax 2M

	(e) Ans.	**Describe with example any two operations on pointers.** **Arithmetic operations which can be done on pointers are** **a) Incrementing a pointer:** Incrementing Pointer Variable Depends Upon data type of the Pointer variable E.g: ptr is an integer pointer and if ptr ++ is the statement executed it will increment the location by 2 because int type needs 2 bytes of storage in 'c' language. Following table shows changes in the address after increment:	**4M** ***Any two operations 2M each***

Data Type	Older Address stored in pointer	Next Address stored in pointer after incrementing (ptr++)
int	1000	1002
float	1000	1004
char	1000	1001

b) Decrementing a pointer : Decrementing of Pointer Variable Depends Upon : data type of the Pointer variable
E.g ptr is an integer pointer and if ptr -- is the statement executed it will decrement the location by 2 because int type needs 2 bytes

of storage in 'c' language.
Following table shows changes in the address after decrement:

Data Type	Older Address stored in pointer	Next Address stored in pointer after incrementing (ptr–)
int	1000	0998
float	1000	0996
char	1000	0999

c) Pointer and a number also can be added or subtracted as follows: If ptr is an integer pointer showing values as 1000 then ptr+3 shows the incremented address as 1006 because it works as address+(number*(size of data type)).
Similarly, for subtracting a number from pointer works as address-(number*(size of data type)).

Relational operations on pointers
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 if statement.

(f) **Define recursive function. List any two advantages of recursive function.** **4M**

Ans.

Definition :
Recursive function is the process in which function calls itself.

Definition 2M

Advantages :
- Reduces length of the program
- Reduces unnecessary calling of a function.
- Useful when same solution is to be applied many times.

Any 2 advantages 1M each

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

(a) **Write a program to sort element of an array descending order.** **4M**

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

		```
		{
		 int arr[5];
		int i,j,temp;
		clrscr();
		printf("\n Enter elements:");
		for(i=0;i<5;i++)
		 {
		  scanf("%d",&arr[i]);
		 }
		for(i=0;i<4;i++)
		 {
		 for(j=i+1;j<5;j++)
		 {
		   if(arr[i]<arr[j])
		 {
		temp=arr[i];
		arr[i]=arr[j];
		arr[j]=temp;
		 }
		}
		}
		printf("\nSorted array elements :\n");
		for(i=0;i<5;i++)
		printf("%d ",arr[i]) ;
		getch();
		 }
		```
			Correct logic 2M
			
			Correct syntax 2M

	(b) Ans.	**Explain '*' and '&' operators used with pointer.** **1.* operator:-** It is used to declare a pointer variable. *Example*: int *ptr; The above statement declares ptr as an integer pointer variable. It is also used as value at operator i.e. it reads the value from the address stored in pointer variable. *Example*: printf("%d", *ptr); The above statement displays value present at the address stored in ptr variable.	**4M** ***Explanation of '*' operator 2M***

		2. & operator:- It is used to retrieve address of a variable from memory. *Example*: int *ptr,a; ptr=&a; The above statement stores the address of variable a in the pointer variable ptr.	***Explanation of '&' operators 2M***
	(c) Ans.	**Explain strlen() and strcmp() string functions with examples.** **strlen()-**This library function is used to count the length of the string i.e. number of characters including blank spaces. ***Syntax* :** strlen(string1); ***Example :*** **int i;** **char string1[]="abc";** i=strlen(string1); strlen function counts number of characters from string1 and stores the count in the variable i. **strcmp( ):**This library function is used to compare two strings which are passed as arguments to it. If the strings are equal then function returns value as 0 and if they are not equal then the function returns ASCII value difference of the first mismatched characters from the string. ***Syntax:*** strcmp(string1,string2); ***Example:*** Consider str1="abc" and str2="abc" i=strcmp(str1,str2) strcmp function compares characters from str1 and str2 and returns 0 as both the strings are same.	**4M** ***Explanation of strlen() 1M, Example 1M*** ***Explanation of strcmp()1M, Example 1M***
	(d) Ans.	**Describe with syntax and example use of continue statement.** **Use:** 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++)	**4M** ***Use of continue 1M*** ***Syntax 1M***

		{
		   if (j==4)
		   {
		      continue;
		   }
		   printf("%d ", j);
		}
		In the above example, printf will display value as 0,1,2,3,5,6,7,8. Value 4 is not displayed because when j=4 continue statement skips printf() statement and continues with next iteration of for.
			Example 2M

	(e)
	
	Ans.
		Write syntax of declaration of function with example.
		(Note: Any Category of function shall be consider)
		Syntax to declare a function:
		Return_type function_name(datatype arg1,datatype arg2,…,datatype arg n)
		{
		Function body;
		}
		
		Example:-
		
		void square(int no)
		{
		printf("%d",no*no);
		}
		In the above example variable no is passed as an argument to the function square.this function displays square of no on screen.
			4M
			
			Syntax 2M
			
			Example 2M

	(f) Ans.	**Explain any two logical operators with example.** **Logical operators:** 1. && Logical AND 2. || || Logical OR 3. ! Logical NOT **1. Logical AND- &&** It is used to test more than one condition. When all conditions are true then the associated statements are executed. *Example:*- if(a>b && a>c) printf("a is greater"); In the above example if both the conditions are true then printf() will	**4M** ***Any two Explanation of each 1M*** ***Example of each***

		display 'a is greater'. If any one of the condition is false then nothing will be displayed. **2. Logical OR- ||** It is used to test more than one condition. When any one of the condition is true then the associated statements are executed. *Example:-* if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u') printf("character is vowel"); In the above example if any one condition is true then printf () statement will display 'character is vowel'. **3. Logical NOT- !** It is used to reverse the logical state of its operand. If a condition is true then NOT operator will make it false. *Example*:- if(!(a==b)) Printf("both are different"); In the above example if a is not equal to b then printf () will display both are different.	***1M***
4.	**(a)** Ans.	**Attempt any FOUR of the following:** **State any four features of pointer.** 1. Pointers are more efficient in handling arrays and data tables. 2. They can be used to return multiple values from a function via function arguments. 3. Pointers permit references to functions and thereby facilitating passing of functions as arguments to other functions. 4. The use of pointer arrays to character strings results in saving of data storage space in memory. 5. Pointers allow C to support dynamic memory management. 6. Pointers reduce length and complexity of programs. 7. They increase the execution speed and thus reduce the program execution time.	**16** **4M** ***Any four 1M each***
	(b) Ans.	**Explain increment and decrement operator with example.** **Increment operator:** Increment operator (++) is a unary operator. It operates on one operand. It is used to add one to an existing value of variable. Syntax: variable_name++ or ++variable_name *Example:* int num=6;	**4M** ***Explanation of increment operator 1M***

		printf("%d",num);
		num++;
		printf("\n%d",num);
		
		In above example initially value of num is 6. Due to increment operator (++) value of variable num will become 7.
		
		Decrement operator:
		Decrement operator(--) is an unary operator. It operates on one operand. It is used to subtract one from an existing value of variable.
		Syntax: variable_name-- or --variable_name
		
		Example:
		int num=5;
		printf("%d",num);
		num--;
		printf("\n%d",num);
		
		In above example initially value of num is 5. Due to decrement operator (--) value of num will become 4.
			Example 1M
			
			Explanation of decrement operator 1M
			Example 1M

	(c)
	
	Ans.
		Write a function to print factorial of a number.
		(Note: Any Category of function shall be consider)
		void factorial()
		{
		int no,fact=1,i;
		printf("Enter number:");
		scanf("%d",&no);
		for(i=1;i<=no;i++)
		fact=fact*i;
		printf("\n Factorial of %d is %d",no,fact);
		}
			4M
			
			Correct function with correct syntax 4M

	(d)
	
	Ans.
		Write a 'c' program to accept any integer number and print whether it is even or odd.
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int num;
		clrscr();
			4M
			
			Correct logic 2M

		printf("Enter a number:");
		scanf("%d",&num);
		if(num%2==0)
		{
		printf("The number %d is even",num);
		}
		else
		{
		printf("The number %d is odd",num);
		}
		getch();
		}
			Correct syntax 2M

	(e) Ans.	**Describe the concept of array of structure with example.** **Array of structure:-** Array of structure means collection of structure variables. It can be used when we want to use many variables of the same structure. *For example:* If a structure for student data is defined and it has to be used for 10 different students then array of structure can be declared as : struct student { int rollno; char name[20]; } s[10]; Here data in the form of rollno and name can be stored or accessed for 10 students. For eg : s[0].rollno and s[0].name will be the data for first student. s[1].rollno and s[1].name will be the data for second student and so on.	**4M** ***Description 2M*** ***Example 2M***

	(f)
	Ans.
		Write a program for addition of two 3 x 3 matrix.
		#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");
			4M
			
			Correct logic 2M

		```
		for(i=0;i<3;i++)
		{
		 for(j=0;j<3;j++)
		{
		 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();
		 }
		```
			Correct syntax 2M

5.	**(a)** Ans.	**Attempt any FOUR of the following:** **Explain formatted input and formatted output with example.** **Formatted input:** When the input data is arranged in a specific format, it is called formatted input. scanf function is used for this purpose in C.	**16** **4M**

		General syntax:
		scanf("control string", arg1, arg2..);
		control string here refers to the format of the input data. It includes the conversion character %, a data type character and an optional number that specifies the field width. It also may contain new line character or tab.
		arg1, arg2 refers to the address of memory locations where the data should be stored.
		Example:
		scanf("%d",&num1);
		
		Formatted output:
		printf is used for formatted output to standard output depending on the format specification. Format specifier, along with the data to be output is the parameters to the function. The different format specifiers used are:
		%d-int values
		%f-float values
		%c-for char values
		%s- for string
		General syntax:
		printf("control string",data1,data2..)
		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);
			Explanation of formatted input 1M
			
			Example 1M
			
			Explanation of formatted output 1M
			
			Example 1M

	(b)
	
	Ans.
		Define following terms: (i) Variable (ii) Identifier (iii) Keyword (iv) Constant
		(i) Variable: it is a data name that may be used to store a data value. The values of variables, unlike constants, can be changed in the program.
		Eg: int num1;
		    num1=10;
		    printf("%d",num1);
		    num1 = 20;
		    printf("%d",num1);
			4M
			
			Correct definition 1M each

		(ii) Identifier: it refers to the names of variables, functions and arrays. These are user defined names and consists of sequence of letters and digits with the letter as the first character. Eg: int num1, num2 **(iii) Keyword**: these are words that have specific meaning and these meanings cannot be changed. They are reserved words. These cannot be used as identifiers in the program. Eg: while, if, switch etc **(iv) Constant**: it is referred to as a fixed value that does not change while the program is under execution. The different constants are: integer constants, real constants, character constants, string constants etc. Eg: 5,5.5,'y',"Hello"	

	(c)
	
	Ans.
		Write a program to find largest of three numbers using nested if else.
		`#include<stdio.h>`
		`#include<conio.h>`
		`void main() {`
		`        int a,b,c;`
		`        clrscr();`
		`        printf("Enter three numbers");`
		`        scanf("%d %d %d",&a,&b,&c);`
		`        if(a>b) {`
		`                if(a>c) {`
		`                        printf("%d is the greatest number",a);`
		`                } else {`
		`                        printf("%d is the greatest number",c);`
		`                }`
		`        }  else if(b>c) {`
		`                printf("%d is the greatest number",b);`
		`        }  else {`
		`                printf("%d is the greatest number",c);`
		`        }`
		`        getch();`
		`}`
			4M
			
			correct logic 2M
			
			correct syntax 2M

	(d) Ans.	**Write a program to accept ten numbers and print average of it.** `#include<stdio.h>`	**4M**

```
#include<conio.h>
void main() {
 int arr[10];
 int i;
 int sum=0,avg=0;
 clrscr();
 for(i = 0;i<10;i++) {
          printf("Enter number");
          scanf("%d",&arr[i]);
 }

 for(i=0;i<10;i++){
          sum=sum+arr[i];
 }
 printf("sum is %d",sum);
 avg = sum/10;
     printf("Average of 10 numbers is %d",avg);
 getch();
}
```

Correct logic 2M

Correct syntax 2M

(e) **Write a 'c' program to declare a structure student with members as roll no, name and mark. Accept and display data for one student.** **4M**

Ans.

```
#include<stdio.h>
#include<conio.h>
void main() {
          struct student{
                    int roll_no;
                    char name[20];
                    int mark;
          } s;
          clrscr();
          printf("Enter the rollno, name and mark of the student");
          scanf("%d %s %d",&s.roll_no,&s.name,&s.mark);
          printf("The rollno of th student is %d",s.roll_no);
          printf("The name of the student is %s",s.name);
          printf("The mark of student is %d",s.mark);
          getch();
}
```

Correct logic 2M

Correct syntax 2M

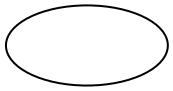
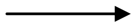
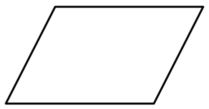
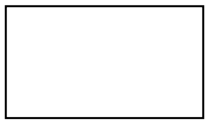
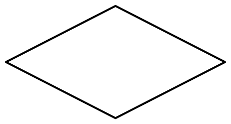
	(f)
	Ans.
		Describe any two storage classes along with suitable example.
		Automatic variables: These are declared inside a function in which they are to be used. They are created when a function is called and destroyed when the function completes its execution. They are private to the function. Therefore these variables are also known as local or internal variables. To declare automatic variables explicitly the keyword auto can be used. The values of automatic variables defined in a function cannot be changed by some other function.
		Eg:
		void main() {
		auto int a;
		a=10;
		printf("%d",a);
		}
		
		External variables: these variables are active and alive throughout the entire program. These are also known as global variables. These variables can be accessed by any function in the program. External variables are declared outside a function. In case a local variable and global variable has the same name, the local variable will have preference over the global variable. The value of a global variable can be changed by any function, the subsequent functions will refer to the new value.
		Eg:
		int number;
		void main() {
		number=10;
		printf("%d",number);
		}
		void function1() {
		number=20;
		printf("%d",number);
		}
		
		Static variables: the value of the static variable persists until the end of the program execution .A variable can be declared as a static using the keyword static. Static variable can be an external or an internal variable. Internal static variable are those who are declared inside a function, but they remain alive throughout the execution of the
			4M
			
			Any 2 Explana tion of each 1M
			
			Exampl e of each 1M

		program. The static variable is initialized only once when the program is compiled.
		Eg:
		`void func1() {`
		` static int x=0;`
		` x= x+1;`
		` printf("x=%d",x);`
		`}`
		`void main() {`
		` int i;`
		` for(i=0;i<3;i++) {`
		`          func1();`
		`         }`
		`}`
		
		Register variables: these variables are stored in the registers instead of memory. Since the register access is much faster compared to the memory, frequently used variables can be stored this way.
		Eg:
		`void main() {`
		`register int count=0;`
		`count++;`
		`printf("%d",count);`
		`}`	

6.	**(a)** Ans.	**Attempt any FOUR of the following:** **Explain the following terms:** **(i) Algorithm** **(ii) Flowchart** ***(Note: Example and symbols are optional)*** **(i) algorithm:** It is a step by step procedure to solve a problem. Starting from the initial step and input it proceeds in a sequential step by step way to get the output. *Eg:* The algorithm to add two numbers can be written in the following way: 1. start 2. declare 3 variables a,b,c; 3. get the input values for the variables a and b 4. calculate the sum c = a+b 5. display the output c	**16** **4M** ***Explanation of algorithm 2M***

6. stop

(ii) Flowchart: It is the diagrammatic representation of methodically solving a problem. It can also be called as the diagrammatic representation of an algorithm.
The different symbols used in flowcharts are:

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

Explanation of flowchart 2M

(b) **Write a program to generate Fibonacci series in C.** **4M**

(Note: Limit for printing the Fibonacci series is upto the students)

Ans.

2M for correct syntax

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int no1,no2,no3,l,i;
 no1 = 0;
```

		```
		no2 = 1;
		clrscr();
		 printf("\n Enter the limit :");
		scanf("%d",&l);

		 printf("%d\n%d\n",no1,no2);
		 for(i=0;i<l-2;i++)
		 {
		no3=no1+no2;
		printf("%d\n",no3);
		no1=no2;
		no2=no3;
		 }
		 getch();
		}
		```
			2M for correct logic

	(c)
	
	Ans.
		Write a program in 'C' to find whether the entered number is positive or negative.
		```
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		 int n;
		 clrscr();
		 printf("Enter a number");
		 scanf("%d",&n);
		 if(n>0)
		          {
		           printf("Number %d is positive",n);
		 }
		          else if(n<0)
		          {
		           printf("Number %d is negative",n);
		 }
		          else
		          {
		           printf("Number is zero");
		 }
		 getch();
		}
		```
			4M
			
			Correct syntax 2M
			
			Correct logic 2M

(d) **Write a program in 'c' to copy one string into other without using built in function.** **4M**

Ans.

```
 #include<stdio.h>
 #include<conio.h>
 void main()
 {
  int i;
  char str[20];
  char dest[20];
  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();
 }
```

Correct logic 2M

Correct syntax 2M

(e) **Declare and initialize the one dimensional array with TEN elements. Explain how the elements in an array can be accessed.** **4M**

Ans. To declare and initialize a one dimensional array with 10 elements:
int arr[10] = {10, 20, 5, 3, 55, 45, 15, 7, 30, 52};
The elements of an array can be accessed by using indices. The first element in an array will be represented by arr[0], the second element arr[1], third element arr[2], fourth element arr[3], fifth element arr[4] so on. The 10th element will be represented by arr[9].

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	arr[7]	arr[8]	arr[9]
10	20	5	3	55	45	15	7	30	52

Declaration and initialization 2M

Elements of an array can also be accessed using loop.
Eg:

```
#include<stdio.h>
#include<conio.h>
void main() {
 int arr[] = {10, 20, 5, 3, 55, 45, 15, 7, 30, 52};
```

Explanation 2M

```
 int i;
 clrscr();
 for(i = 0; i < 10; i++) {
 printf("%d\t",arr[i]);
 }
 getch();
}
```

(f) **Distinguish between call by value and call by reference methods for calling function.** **4M**

Ans.

Call by value	Call by reference
A copy of actual arguments is passed to respective formal arguments.	The location, that is, the address of actual arguments is passed to formal arguments
Actual arguments will remain safe, they cannot be modified accidentally.	Alteration to actual arguments is possible within from called function; therefore the code must handle arguments carefully else you get unexpected results.
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 are reflected outside also.

Any 4 differences 1M each