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

WINTER – 2016 EXAMINATION

Model Answer **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:**
		Give the syntax of switch case statements.
		switch (integer expression)
		{
		case constant 1:
		do this;
		case constant 2:
		do this;
		case constant 3:
		do this;
		default:
		do this;
		}
			20
			2M
			
			Correct syntax- 2M

	b) Ans.	**State four arithmetic and four logical operators.** **Arithmetic operators** 1. + Addition 2. –subtraction 3. *multiplication 4. /division 5. %modular division	***2M*** ***arithmetic operators-1M,***

		Logical operators 1. & Logical AND 2. || Logical OR 3. ! Logical NOT	***logical operators-1M***
	c) Ans.	**Define array. How array is declared write its syntax?** **Definition**: An array is a collection of similar type of elements. **Syntax**: data_type array_variable_name[size];	***2M*** ***Definition-1M, syntax for declaration-1M***
	d) Ans.	**Define token and identifier.** **Token:** In a program, the smallest individual unit is known as Token. e.g. keyword, constants **Identifier:** Identifier is a user-defined name and consists of a sequence of letters and digits. It refers to the names of variables, functions and arrays. e. g. main, amount	***2M Correct definition of token-1M*** ***identifier-1M***
	e) Ans.	**Define function.** A function is a self-contained block of code that performs a particular task i.e. it is a collection of statements to perform a particular task.	***2M Correct definition 2M***
	f) Ans.	**State any four control statements.** Control statements:- 1. if 2. if-else 3. break 4. continue 5. switch 6. goto 7. while 8. for	***2M*** ***Any four-each ½ M***
	g) Ans.	**Define recursive function.** Recursion means a function calls itself repetitively. A recursive function contains a function call to itself inside its body.	***2M Correct definition-2M***
	h) Ans.	**State the use of break and continue statement.**	***2M***

<table>
<tr><td></td><td></td><td>break :
The break statement is used to break the control in the loops. When break statement is executed inside any loop, control automatically passes to the first statement after the loop.

continue:
The continue statement is used to transfer the control in the beginning of the loop. Whenever continue statement is executed inside any loop automatically control passes to the beginning of the loop.</td><td>Use of break- 1M

continue -1M</td></tr>
<tr><td></td><td>i)
Ans.</td><td>State the use and syntax of strcmp ( ) function.
Use: - This function is used to compare two strings. It compares two strings and returns zero if both strings are equal, otherwise it returns the difference between ASCII values of first non matching character pair from the strings.
Syntax: strcmp (string1, string2);</td><td>2M

Use-1M

syntax- 1M</td></tr>
<tr><td></td><td>j)
Ans.</td><td>Define pointer. How pointer is declared?
Definition:
A pointer is a variable that stores memory address of another variable which is of similar data type.

Declaration:
datatype *pointer_variable_name;</td><td>2M

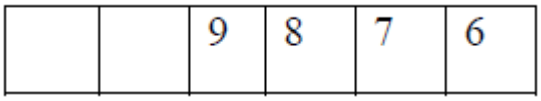
Correct definitio n-1M

Declarat ion-1M</td></tr>
<tr><td></td><td>k)
Ans.</td><td>State difference between array and string.

<table>
<tr><th>Sr. No.</th><th>Array</th><th>String</th></tr>
<tr><td>1</td><td>An array is a fixed size sequenced collection of similar type of elements.</td><td>A string is a sequenced collection of characters.</td></tr>
<tr><td>2</td><td>The last element of an array is an element of the specific type.</td><td>The last character of a string is a ‘\0’ character.</td></tr>
<tr><td>3</td><td>Syntax:
data_type variable_name[size];</td><td>Syntax:
char variable_name[size];</td></tr>
<tr><td>4</td><td>Example:
int a[5];</td><td>Example:
char name[10];</td></tr>
</table>
</td><td>2M

Any two differen ces: each 1M</td></tr>
<tr><td></td><td>l)</td><td>Write output of the following program:
Void main ( )</td><td>2M</td></tr>
</table>

	Ans.	**{** **int sub[10], i;** **for (i=0; i<=8; i++)** **{** **sub [i] = i;** **printf ("\n %d", sub [i]);** **}** **}** ***Output:*** 0 1 2 3 4 5 6 7 8	***Correct output- 2M***
2	**a)** Ans.	**Attempt any FOUR of the following:** **Explain formatted input and formatted output statements.** **Formatted input:** **Formatted input** refers to an input data that has been arranged in a particular format. The format of the input i.e. scanf ( ) function includes: Format specification, consisting of the conversion character %, a data type character and an optional number specifying the field width. Example: scanf ("%2d", &num1); For the above statement suppose input number is 31426 then by %2d it will only take 31 from 31426. **Formatted Output:** Formatted output refers to an output data that has been arranged in a particular format. The format of the output i.e. printf ( ) function includes: 1. Characters 2. Format specification 3. Escape sequence characters such as \n, \t and \b. Example: printf ("%6d",9876);	**16** ***4M*** ***Explanation of formatted input- 2M*** ***Explanation of formatted output- 2M***

		 In the above example 6 is the field width. Hence for the number 9876 two leading spaces will appear in the output.	

	b)
	Ans.
		Write a program to find whether the character entered through keyboard is a vowel or consonant.
		(Note: Any other logic shall be considered)
		#include<stdio.h>
		void main()
		{
		char ch;
		printf("Enter the character");
		scanf("%c",&ch);
		if(ch=='A'||ch=='E'||ch==' I ' || ch=='O' ||ch=='U'|| ch=='a' ||ch=='e' ||ch=='i'||ch=='o'||ch=='u')
		printf("\n Entered character is Vowel");
		else
		printf("\n Entered character is consonant");
		}
			4M
			Correct logic- 2M, Syntax 2M

	c)
	Ans.
		Write a program in C to find maximum of three nos.
		(Note: Any other logic shall be considered)
		#include<stdio.h>
		void main()
		{
		int no1,no2,no3;
		printf("\n Enter three numbers:");
		scanf("%d%d%d",&no1,&no2,&no3);
		if(no1>no2)
		printf("\n no1 is maximum");
		else if(no2>no3)
		printf("\n no2 is maximum");
		else
		printf("\n no3 is maximum");
		}
			4M
			Correct logic- 2M, Syntax 2M

	d) Ans.	**Explain the syntax and example strlen ( ) and strcpy ( ) functions.** **strlen()**-This library function is used to count the length of the string i.e. number of characters including blank spaces. ***Syntax :*** strlen(string1);	***4 M Explanation of Syntax***

		Example : i=strlen(st1); strlen function counts number of characters from st1 and returns the result in the variable i. **strcpy( ):-**This library function is used to copy the content of one string to the other string. ***Syntax***: strcpy(destination string ,source string); ***Example***: strcpy(string1,string2); strcpy function copy contents of string 2 into string 1.	***and example of strlen-2 M*** ***strcpy-2M***
	e) Ans.	**Explain '*' and '&' operators used in pointers.** **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. **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'.	***4M*** ***Explanation of * operator 2M*** ***Explanation of & operator 2M***

	f)
	
	Ans.
		Write a program to calculate factorial of number using function.
		(Note: Any type of function shall be considered)
		#include<stdio.h>
		void factorial(int no);
		void main()
		{
		int no;
		printf("Enter number");
		scanf("%d",&no);
		factorial(no);
		}
		void factorial(int no)
		{
			4M
			
			Correct logic-2M, Syntax 2M

		int fact=1,i;
		for(i=1;i<=no;i++)
		fact=fact*i;
		printf("\n Factorial of %d is %d",no,fact);
		}	

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

	a)
	Ans.
		Explain while loop with syntax and example.
		The while is an entry – controlled loop statement. The test- condition is evaluated and if the condition is true, then the body of the loop is executed. After execution of the body, the test condition is once again evaluated and if it is true, the body is executed once again. The process of repeated execution of the body continues until the test condition finally becomes false and the control is transferred out of the loop. On exit, the program continues with the statement immediately after the body of the loop.
		Syntax:
		while(test condition)
		{
		Body of the loop
		}
		Example :
		main()
		{
		int i=1;
		while(i <=10)
		{
		printf("%d ",i);
		i++;
		}
		}
		This will produce the output as 1 2 3 4 5 6 7 8 9 10
			4M
			Explanation 2M
			Syntax 1M
			Example 1M

	b) Ans.	**Explain with example array of pointer.** A pointer is a variable that contains an address which is a location of another variable in memory. Syntax to create an array of pointers: data type * arr_name[size]; ***Example:*** int *x[2]; char *name[3]={"Nashik","Mumbai","Pune"}; **Program:** #include<stdio.h>	***4M*** ***Explanation 2M*** ***Example or***

		#include<conio.h>
		void main( )
		{
		char *str[5]={"keyboard", "monitor", "CPU","harddisk", "SMPS"};
		int i =0;
		clrscr( );
		for(i=0;i<=4;i++)
		printf("\n %d element = %s",i,str[i]);
		getch();
		}
			program 2M

	c)
	
	Ans.
		State the use of increment and decrement operator. Also write the difference between i++ and ++i.
		increment operator ++ :
		The operator ++ adds 1 to the operand. It is unary operator which can be used on a single operand. Example: ++a, b++;
		decrement operator --:
		The operator – subtracts 1 from the operand. It is unary operator, which can be used on a single operand.
		Example : --p, m--;
		Difference between i++ and ++i:
		1) When ++ operator is used as suffix to the operand then it is called as postfix operator and works as 1) assignment first and 2) then addition of 1.
		Example :
		int i=5;
		int j;
		j=i++;
		Then 1) j=i and then 2) i=i+1 will be the sequence of operations, Output will be j=5 and i=6.
		2) When ++ operator is used as prefix to the operand then it is called as prefix operator and works as 1) addition of 1 first and 2) then assignment.
		Example :
		int i=5;
		int j;
		j=++i;
		Then 1) i=i+1 and then 2) j=i will be the sequence operations, Output will be i=6 and j=6.
			4M
			
			Use 2M
			
			Differen ce 2M

	d) Ans.	**Explain in brief auto and extern storage class.** **auto:** Automatic variables are generally declared in the function in which they	***4M***

		get utilized. They are known as local variable. Automatic variable can be created when function is called and get destroyed when execution of function gets over. We can declare variable with auto as follows:
		main( )
		{
		auto int number;
		…….
		…….
		}
		extern:
		External variables are global variable which are declared above the main ( ) function. Since they are global, they are accessible to all the functions in the program. Scope of these variables is throughout the program.
		extern int a;//declarations
		main()
		{
		……..
		……..
		}
			Explanation of auto 2M
			
			Explanation of extern 2M

	e) Ans.	**With suitable example, explain how else if ladder can be used.**
		else if ladder is used to take a multipath decision. It is used in a program when there are more than one conditions are involved.
		The conditions are evaluated from the top to the bottom. As soon as the true condition is found, the statement associated with it is executed and the control is transferred to the statement-x. When all the conditions become false, then the final else containing the default statement will be executed.
		Example:
		#include<stdio.h>
		void main()
		{
		int no1,no2,no3;
		printf("\n Enter three numbers:");
		scanf("%d%d%d",&no1,&no2,&no3);
		if(no1>no2)
		printf("\n no1 is greater");
		else if(no2>no3)
		printf("\n no2 is greater");
		else
		printf("\n no3 is greater");
		}
			4M
			
			Explanation 2M
			
			Example 2M

		Explanation w.r.t. example : In above example first if compares two numbers no1 and no2, if no1 is greater, prints the result otherwise it goes to the next condition to check, where no2 and no3 are compared for maximum , if not the second, it prints that the third number is maximum. Here else if ladder is used because if first condition is false, then only second condition is checked, if it is false then third and so on.	

	f)
	
	Ans.
		Write a program to declare structure book having data member as book_name, bookid, book_price. Accept this data for 3 books and display it.
		#include<stdio.h>
		main()
		{
		int i;
		struct book
		{
		char book_name[20];
		int bookid;
		int book_price;
		}b[3];
		for(i=0;i<3;i++)
		{
		printf("Enter details for book %d :",i+1);
		scanf("%s %d %d", b[i].book_name,&b[i].bookid,&b[i].book_price);
		}
		printf("Details of books :\n");
		for(i=0;i<3;i++)
		{
		printf("%s %d %d\n", b[i].book_name,b[i].bookid,b[i].book_price);
		}
		}
			4M
			
			Correct logic- 2M
			
			Syntax 2M

4		**Attempt any <u>FOUR</u> of the following:**	**16**
	a) Ans.	**With suitable example, explain how a structure can be initialized.** A structure variable can be initialized at compile time. Compile time initialization of a structure variable must have following elements : 1) Keyword struct. 2) Structure tag name. 3) Name of the variable 4) Assignment operator =	***4M*** ***Explanation 2M***

		5) Set of values for the members of structure variable, separated by commas and enclosed in braces.
		6) Terminating semicolon.
		Example :
		Case a)
		main()
		{
		struct student
		{
		int rollno;
		int marks;
		};
		struct student s1={101,78};
		struct student s2={102,89};
		….
		….
		}
		
		OR
		Case b)
		main()
		{
		struct student
		{
		int rollno;
		int marks;
		}student s1={101,78};
		
		struct student s2={102,89};
		….
		….
		}
		Explanation w.r.t. example :
		In above example, variables 's1' and 's2' are structure variables and are initialized by assigning two values as rollno and marks separated by comma, enclosed with braces. Either it can be initialized at the time of structure declaration as in case (b) or later on as in case (a).
			Example 2M

	b)	**Explain meaning of following statement with reference to pointers.** **int *a, b;**	***4M***

b = 20;
***a = b;**
a = &b;

Ans.

1) int *a, b;
It is declaring a basic int type variable 'b' and 'a' is declared as pointer to integer.
2) b = 20;
It is storing value 20 to int variable 'b'.
3) *a = b;
It is storing value of 'b' at address stored in pointer 'a'.
4) a = &b;
It is storing address of 'b' into pointer variable 'a'.

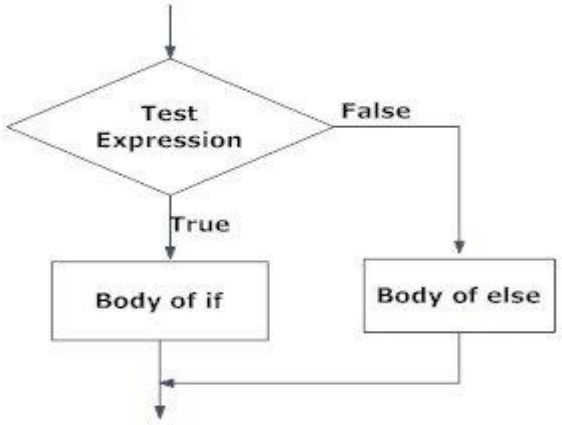
Each statement 1M

c) Ans. **Write difference between call by value and call by reference.** ***4M***

call by value	**call by reference**
1) In call by value, a copy of actual argument is passed to formal arguments of the called function	1) In call by reference the location (address) of actual argument is passed to formal arguments of called function.
2) Any changes made to the formal arguments in called function have no effect on the values of actual argument	2) Any changes made to the formal argument in called function affects the values of actual argument
3) Pointers are not used	3) Pointers are used

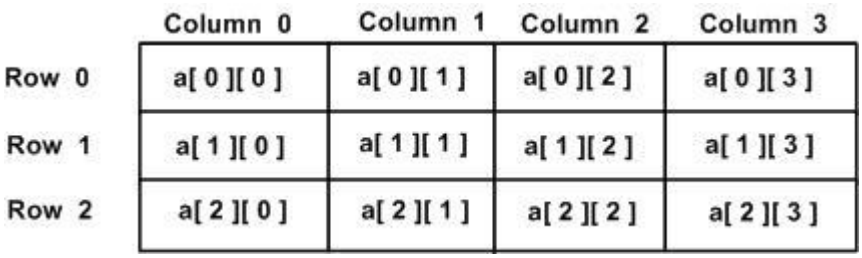
4) *Example*:
#include<stdio.h>
void swapbyvalue(int,int);
int main()
{
int n1=10, n2=20;
swapbyvalue(n1,n2);
printf("n1=%d,n2=%d",n1,n2);
}
void swapbyvalue(int a, int b)
{
int t;
t=a; a=b; b=t;
}
	4) *Example*:
	#include<stdio.h>
	void swapbyreference(int *, int *);
	int main()
	{
	int n1=10, n2=20;
	swapbyreference(&n1,&n2);
	printf("n1=%d,n2=%d",n1,n2);
	}
	void swapbyreference(int * a, int* b)
	{
	int t;
	t=*a; a=*b; *b=t;
	}

Any four differences 1M each

	d)
	
	Ans.
		Explain use of if else statement. Also draw the flow chart for the same.
		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)
		{
		True-block statement (s)
		}
		else
		{
		False-block statement (s)
		}
		Statement-x;
		
		Flow Chart :
		 
		
			4M
			
			Explanation 3M
			
			Flow chart 1M

	e) Ans.	**Explain conditional operator with example.** **Conditional Operator (Ternary Operator):** It takes the form "?:" to construct conditional expressions. The operator "? :" works as follows: Syntax: exp1? exp2 : exp 3 ; Where exp1, exp2 and exp3 are expressions. exp1 is evaluated first, If it is true, then expression exp2 is evaluated. If exp1 is false, exp3 is evaluated.	***4M*** ***Explanation 3M***

		***Example*:** int a=10,b=5,x; x=(a>b) ? a : b; In the above example x will take value 10 because condition given is if a>b.	***Example 1M***

	f)
	Ans.
		With suitable example, explain how two dimensional arrays can be created.
		Two dimensional arrays can be declared as
		data_type arrayname[row_size][col_size];
		Two dimensional arrays are stored in memory as shown in figure below. Each dimension of the array starts with zero. The first index indicates row and second index indicates column.
		E.g. int a[3][4] appears as follows :
		 
		
		Example :
		main()
		{
		Int i, j;
		Int a[3][4];
		for(i=0;i<3;i++) // row
		{
		for(j=0;j<4;j++) // column
		{
		scanf("%d",&a[i][j]);
		}
		}
			4M
			Explanation 3M
			Example 1M

5	**a)** Ans.	**Attempt any <u>FOUR</u> of the following:** **Write a 'C' program to accept two integer Nos from user and print the result.** #include<stdio.h> #include<conio.h> void main() { int num1, num2;	**16** ***4M*** ***Correct syntax 2M, correct logic 2M***

		clrscr();
		printf("Enter 2 numbers");
		scanf("%d%d",&num1, &num2);
		printf("The numbers are %d %d",num1,num2);
		getch();
		}	

	b)
	
	Ans.
		Write a program to find whether a entered number is even or odd.
		(Note: Any other logic shall be considered)
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int num1;
		clrscr();
		printf("Enter a number");
		scanf("%d",&num1);
		if(num1%2==0)
		{
		printf("The number %d is even",num1);
		}
		else
		{
		printf("The number %d is odd",num1);
		}
		getch();
		}
			4M
			
			Correct syntax 2M, correct logic 2M .

	c) Ans.	**What is function prototype? Explain with example.** ***(Any relevant example may be considered)*** The execution of a C program starts from main ( ) function. Any other function defined, should be called from the main ( ). Like variable declaration, a function should be declared before using it in a program. In C, declaration of a function is called function prototype. Function prototype declaration gives the information regarding the name of the function, return type of the function, parameter list and name of the function to the compiler. Function prototype ends with a semicolon. return_type name(parameterlist); *Example:* #include<stdio.h>	***4M*** ***Explanation 2M***

```
#include<conio.h>
void findEven( );
void main( )
{
int num;
clrscr();
printf("Enter a number");
scanf("%d",&num);
findEven(num);
getch();
}
void findEven(int n)
{
  if(n%2==0)
  {
   printf("Number is even");
  }
else
  {
   printf("Number is odd");
  }
}
```

Example 2M

d) **What is array? How elements of single dimensional array can be accessed.** — ***4M***

Ans.

An array is a data structure which can hold a number of values of the same data type. The values in an array are stored in continuous memory locations. — ***Definition of array- 1M***

To declare and initialize an array:

datatype arr_name[size] = {val1, val2, val3, val4 ...valn};

int arr[5] = {10, 20, 5, 3, 55};

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. — ***Explanation of accessing elements -3M***

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
10	20	5	3	55

Elements of an array can also be accessed using loop.

Example:

```
#include<stdio.h>
```

```
#include<conio.h>
void main()
{
int arr[] = {5, 34, 6, 23, 55};
int i;
clrscr();
for(i = 0; i < 5; i++) {
printf("%d\t",arr[i]);
}
getch();
}
```

e) Ans.

Explain special operators in 'C' with example. *4M*

Operators	Description
&	This is used to get the address of the variable. Example : &a will give address of a.
*	This is used as pointer to a variable. Example : * a where, * is pointer to the variable a.
Sizeof ()	This gives the size of the variable. Example : size of (char) will give us 1.
Comma(,)	This can be used to link the related expressions together. A comma linked expression is evaluated from left to right and the value of the right most expression is the value of the combined expression. x = (a = 2, b = 4, a+b)
Dot(.)	This is used tom access a structure member Structurevariable.variablename Student.stud_id
arrow(->)	The address of a structure variable can be stored in a pointer variable and the individual members of the structure can then be accessed using the pointer variable and the **Arrow Operator -> .** struct student student1; struct student *stud1; *stud1=&student1; stud1->student_id;

Any four - 1M each

f) Ans.

Write a program using loop to print following: *4M*

1
2 3
4 5 6

```
#include<stdio.h>
#include<conio.h>
```

		void main() {
		int i, j, k;
		k=1;
		clrscr();
		for(i=0;i<3; i++)
		{
		for(j = 0; j <= i; j++)
		{
		printf("%d\t",k);
		k++;
		}
		printf("\n");
		}
		getch();
		}
			Correct syntax 2M, correct logic – 2M

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

	a) Ans.	**Write a program to print even numbers between 1 to 100.**
		#include<stdio.h>
		#include<conio.h>
		void main( )
		{
		int i;
		clrscr( );
		i=2;
		while(i<100)
		{
		printf("%d\n",i);
		i=i+2;
		}
		getch( );
		}
			4M
			
			Correct syntax 2M correct logic 2M

	b) Ans.	**List different categories of function and explain any one in detail.** The different categories of functions are: 1. Function without arguments without return type 2. Function without arguments with return type 3. Function with arguments without return type 4. Function with arguments with return type **Function without arguments without return type**: Here the function will not return any value and it will not have any argument. *Example:* #include<stdio.h>	***4M*** ***Listing of categories 2M***

		#include<conio.h>
		void printNum();
		void main()
		{
		printNum();
		getch();
		}
		void printNum()
		{
		int i = 10;
		printf("%d",i);
		}
		Function without arguments with return type: Here the function will return a value from the function but it will not have any arguments.
		Example:
		#include<stdio.h>
		#include<conio.h>
		int printNum();
		void main()
		{
		int i = printNum();
		printf("%d",i);
		getch();
		}
		int printNum()
		{
		int i = 10;
		clrscr();
		return i;
		}
		Function with argument without return type: Here the function takes values as arguments but it does not return any value
		Example:
		#include<stdio.h>
		#include<conio.h>
		void printNum(int);
		void main()
		{
		int i = 10;
		clrscr();
			Explanation of any one -2M

```
  printNum(i);
  getch();
}
void printNum(int i)
{
printf("%d",i);
}
```

Function with arguments with return type: Here the function takes values as arguments and returns value.

Example:

```
#include<stdio.h>
#include<conio.h>
int printNum(int);
void main()
{
 int i = 10;
 int sq=0;
 clrscr();
 sq = printNum(i);
 printf("%d",sq);
 getch();
}
int printNum(int i)
{
  int s = i*i;
 return s;
}
```

c) Ans. **Explain concept of array of character with example.** — ***4M***

An array is a data structure which holds different values. Each value will be stored in continuous memory locations. Each element in an array can be accessed using the indices. An array of characters holds different characters. It can be declared and initialized as:

char arr[] = {'c','o','l','l','e','g','e','\0'};

where - arr[0] = 'c', arr[1] = 'o' etc.

Since an array of characters contains sequence of characters it is also called a string. A null character is inserted into the end of a string.

An array of characters can be also declared in the following way:

char arr[] = "college";

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	arr[7]
c	o	l	l	e	g	e	\0

Concept and explanation 3M

		Example:
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		char arr[] = {'h','e','l','l','o','\0'};
		int i;
		clrscr();
		for(i = 0; i < 6; i++)
		{
		printf("%c",arr[i]);
		}
		getch();
		}
		OR
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		char arr[] = "hello";
		clrscr();
		printf("%s",arr);
		getch();
		}
			Example 1M

	d)
	
	Ans.
		Write a program to determine whether the entered string is palindrome or not.
		(Note: Any other logic shall be considered)
		#include<stdio.h>
		#include<conio.h>
		void isPalindrome(char str[])
		{
		int l = 0;
		int h = strlen(str) -1;
		while(h>l)
		{
		if(str[l++] != str[h--])
		{
		printf("%s is not palindrome",str);
		return;
			4M
			
			Correct syntax 2M, correct logic 2M

		}
		}
		printf("%s is palindrome",str);
		}
		void main()
		{
		char arr[20];
		clrscr();
		printf("Enter a string");
		scanf("%s",&arr);
		isPalindrome(arr);
		getch();
		}	

	e)
	
	Ans.
		Explain with the proper syntax:
		(i) Function definition
		(ii) Function body
		(iii) Function call
		(iv) Function prototype
		(i) Function definition:
		A function definition in C programming consists of a function header and a function body.
		Here are all the parts of a function −
		Return Type − A function may return a value. The return_type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return type is the keyword void.
		Function Name − This is the actual name of the function. The function name and the parameter list together constitute the function signature.
		Parameters − A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.
		Function Body − The function body contains a collection of statements that define what the function does.
		General syntax:
		Return_type name(parameter list)
		{
		body
		}
			4M
			
			Explanation ½M and syntax ½ M for each

		(ii) Function body:
		It contains a collection of statements that define the functionality of the function, ie the body says what the function does.
		void printNumber()
		{
		int i = 10, j = 20;
		int sum = 0;
		sum = i+j;
		printf("%d",sum);
		}
		(iii) Function call:
		Once the function is defined, to execute the function it should be invoked in the main. This process is called calling a function.
		void main()
		{
		printNumber();
		}
		(iv) Function prototype:
		A function should be declared before using it in a program. In C, declaration of a function is called function prototype.
		void printNumber();	

	f)
	
	Ans.
		Write the difference between while and do while loop, and also explain the syntax of it.
		while is an entry controlled loop, that is, the condition is checked before executing any statements of the while loop. The statements inside the while loop is executed only if the condition is true.
		
		do-while is an exit controlled loop, that is, the condition is checked after the statement is executed. The statements inside the do-while loop get executed at least once. do-while loop is used when the statements inside the loop is to be executed at least once even if the condition is false at the first iteration itself. The statements get executed anyway because the condition is checked afterwards.
		
		Syntax of while:
		while(condition)
		{
		//body
		}
		
		Syntax of do-while:
			4M
			
			Difference between while and do-while 2M
			
			Explanation of syntax 2M

		do {
		//body
		} while(condition);
		
		Example:
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int i = 10;
		clrscr();
		while(i>10)
		{
		printf("%d",i);
		i--;
		}
		getch();
		}
		Here in the above program the value of i will not be printed because the condition is false at the entry point itself.
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int i =10;
		clrscr();
		do
		{
		printf("%d",i);
		i--;
		}
		while(i>10);
		getch();
		}
		Here the value of i as 10 will be printed, this is because after the print statement the condition is checked. Thereafter since the condition is false, the loop will not repeat.	