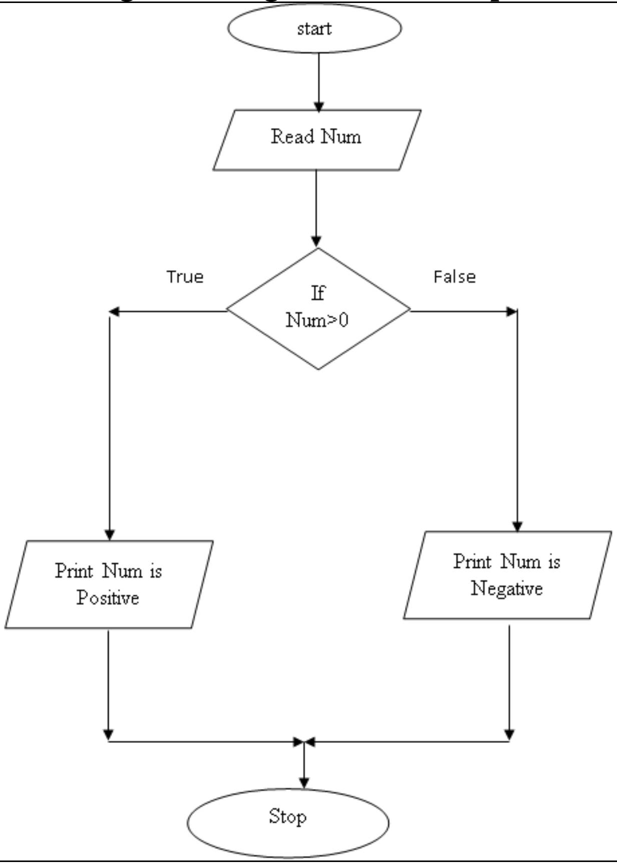
MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2013 Certified)

WINTER – 2023 EXAMINATION
Model Answer – Only for the Use of RAC Assessors

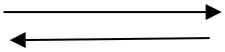
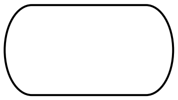
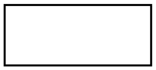
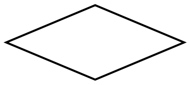
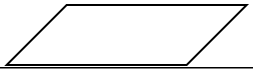
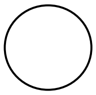
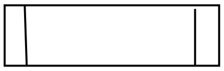
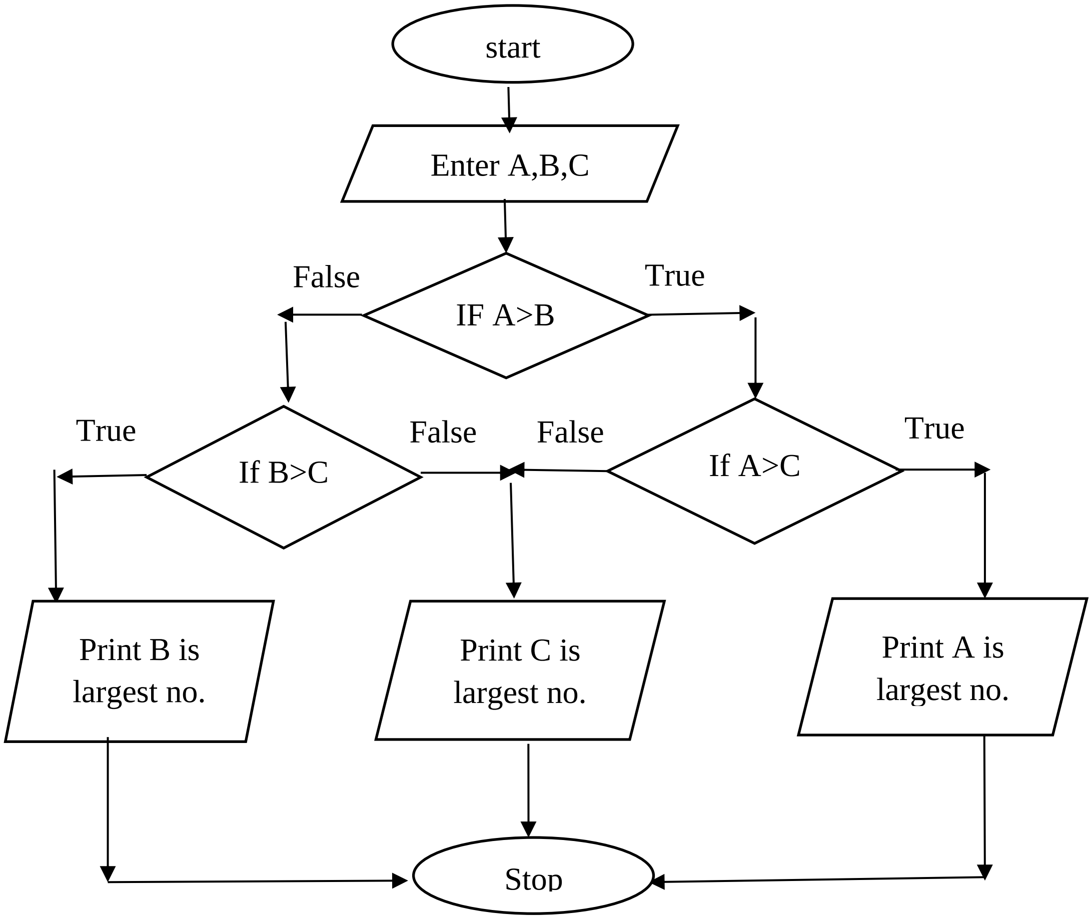
Subject Name: Programming in "C" **Subject Code:** 22226

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.
8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students write answers in Marathi or bilingual language (English +Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1		**Attempt any FIVE of the following:**	**10 M**
	a	**Draw flowchart for checking whether given number is positive or negative.**	**2 M**
	Ans	 start Read Num True If Num>0 False Print Num is Positive Print Num is Negative Stop	Correct sequence 1M Correct symbol 1M

	b	**list any four keywords used in 'C'.**	**2 M**
	Ans	There are 32 keywords in C programming language. The C keywords list are: Auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while.	Any 4 keywords 1/2 M each
	c	**State any two decision making statements.**	**2 M**
	Ans	Decision making statements: 1. if statement 2. if-else statement 3. if-else-if ladder 4. Nested if-else statement 5. switch statement 6. conditional operator statement (? : operator)	Any two statement 1M each
	d	**Define array and list its type.**	**2 M**
	Ans	An array is a collection of data items, all of the same type, accessed using a common name. **OR** Array is a fixed-size sequential collection of elements of the same type. **OR** An array is defined as the collection of similar type of data items stored at contiguous memory locations. **Types:** 1. One dimensional array 2. Two dimensional array 3. Multi-dimensional array	Definition 1M Types 1M
	e	**List the categories of user defined function.**	**2 M**
	Ans	There are four types of user-defined functions: 1. Function with no arguments and no return type. 2. Function with no arguments and a return type. 3. Function with arguments and no return type. 4. Function with arguments and with return type.	1/2 M Each function
	f	**Define pointer. Write syntax for pointer declaration.**	**2 M**
	Ans	**Definition:** A pointer is a variable that stores memory address of another variable which is of similar data type. **Declaration:** datatype *pointer_variable_name; **Eg**: int *ptr;	1M Definition and 1 M syntax

<table>
<tr><td></td><td><b>g</b></td><td><b>Draw and label any four symbols used in flowcharts.</b></td><td><b>2 M</b></td></tr>
<tr><td></td><td><b>Ans</b></td><td>
<table>
<tr><th>Symbol</th><th>Symbol Name</th></tr>
<tr><td></td><td><b>Flow Lines</b></td></tr>
<tr><td></td><td><b>Terminal</b></td></tr>
<tr><td></td><td><b>Processing</b></td></tr>
<tr><td></td><td><b>Decision</b></td></tr>
<tr><td></td><td><b>Input/ Output</b></td></tr>
<tr><td></td><td><b>Connector</b></td></tr>
<tr><td></td><td><b>Predefined Process</b></td></tr>
</table>
</td><td>Any 4 symbols 1/2 M each</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td><b>2</b></td><td></td><td><b>Attempt any THREE of the following:</b></td><td><b>12 M</b></td></tr>
<tr><td></td><td><b>a</b></td><td><b>Draw flowchart for finding largest number among three numbers.</b></td><td><b>4 M</b></td></tr>
<tr><td></td><td><b>Ans</b></td><td>


</td><td>Correct sequence 2M
Correct symbol 2M</td></tr>
</table>

b **Write a program using switch statement to check whether entered character is VOWEL or CONSONANT.** **4 M**

Ans **(Any other relevant logic should be considered)**

```
# include<stdio.h>
#include<conio.h>
void main()
{
char ch;
printf(“Enter any character”);
scanf(“%c”, &ch);
switch(ch)
{
case ‘a’:
printf("%c is a vowel",ch);
break;
case 'e':
printf("%c is a vowel",ch);
break;
case 'i':
printf("%c is a vowel",ch);
break;
case 'o':
printf("%c is a vowel",ch);
break;
case 'u':
printf("%c is a vowel",ch);
break;
default:
printf("%c is a consonant",ch);
break;
}
getch();
}
```

Correct logic 3M
Correct syntax 1M

c **Differentiate between character array and integer array with respect to size and initialization.** **4 M**

Ans

Parameter	Character Array	Integer Array
Size	Store individual characters, each occupying one byte of memory. The size of a character array is determined by the number of characters it can hold. For example, a character array of size	Store integer values, which typically occupy multiple bytes depending on the programming language and architecture. The size of an integer array is also determined by the number of integer values it can

Each parameter 2M

	5 can accommodate five characters.	accommodate. For example, an integer array of size 5 can hold five integer values.

Initialization	Initialization can be done like :
	char str[4]={'x','y','z','\0'};
	OR
	char message[10] = "Hello";
		Initialization can be done like :
		int arr[4]={10,20,30,40};

d	**Explain pointer arithmetic operations with example.**	**4 M**

Ans

Any two operators Each operator with explanation 1M

1M for each example

Pointer Arithmetic is the set of arithmetic operations that can be performed on pointers. The basic operations on pointers are:

1.Increment

Increment is used to increment the pointer. Each time a pointer is incremented, it points to the next memory location.

Example

For an int pointer variable, if the current position of pointer is 1000, when incremented it points to 1002(memory location) because for storing an int value it takes 2 bytes of memory.

```
#include <stdio.h>
#include<conio.h>
void main()
{
   int a = 13;
   int *p = &a;
   printf("p = %u\n", p);
   p++;
   printf("p++ = %u\n", p);
   getch();
}
```

2.Decrement

Decrement is used to decrement the pointer. Each time a pointer is decremented, it points to the previous memory location.

Example

If the current position of pointer is 1002, then decrement operation results in the pointer pointing to the location 1000.

```
#include <stdio.h>
#include<conio.h>
void main()
{
   int a = 22;
  int *p = &a;
   printf("p = %u\n", p);
   p--;
```

```
    printf("p-- = %u\n", p);
    getch();
}
```

3.Addition:

When addition operation is performed on the pointer variable, it shows that particular location in the memory. When a pointer is added with an integer value, the value is first multiplied by the size of the data type and then added to the pointer.

Example

```
#include<stdio.h>
#include<conio.h>
void main()
{
int number=20;
int *p;
p=&number;
printf("Address of p variable is %u \n",p);
p=p+3;
printf("After adding 3: Address of p variable is %u \n",p);
getch();
}
```

4.Subtraction:

When subtraction operation is performed on the pointer variable, it shows that particular location in the memory. When a pointer is subtracted with an integer value, the value is first multiplied by the size of the data type and then subtracted from the pointer.

Example

```
#include<stdio.h>
#include<conio.h>
void main()
{
int number=10;
int *p;
p=&number;
printf("Address of p variable is %u \n",p);
p=p-3;
printf("After subtracting 3: Address of p variable is %u \n",p);
getch();
}
```

3		**Attempt any THREE of the following:**	**12 M**
	a	**State the use of printf() and scanf() with suitable example.**	**4 M**

	Ans	**printf( ) & scanf( ):**
		printf() and scanf() functions are library functions in C programming language defined in “stdio.h” header file.
		Uses of printf():
			Use of printf 1M, Use of scanf 1M

		1. It is used to print the any message onto the output screen.
		2. It is used to print formatted text and values to the standard output stream.
		Uses of scanf():
		1. It is used to read data from keyboard or console.
		2. It is a function that stands for Scan Formatted String. It is used to read data from stdin (standard input stream i.e. usually keyboard).
		Format Specifiers used in printf() and scanf():
		1) %d is used to accept or print integer variable/value.
		2) %c is used to accept or print character variable/value.
		3) %f used to accept or print float variable/value.
		Example:
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int i;
		clrscr();
		printf("Enter a number");
		scanf("%d",&i);
		printf("Entered number is: %d",i);
		getch();
		}
			Example 1M each

	b	**Explain any two string handling functions with syntax and example.**	**4 M**

	Ans	**(Any other relevant example should be considered.)**
		1.strlen():
		strlen() function gives the length of the given string. strlen( ) function counts the number of characters in a given string and returns the integer value. It stops counting the character when null character is found.
		Syntax:
		strlen(stringname);
		Example:
		#include<stdio.h>
		#include<string.h>
		int main()
		{
		char str[] = "Hello";
		int length = strlen(str); // length will be 5
		printf("Length of the string: %d", length);
		return 0;
		}
		Output: Length of the string: 5
		2. strcat():
		strcat() function concatenates (joins) two strings.
		Syntax:
			Explanation with syntax 1M
			Example 1M
			each
			(Any two string functions)

strcat(destination string, source string);

It concatenates source string at the end of destination string.

Example:

```
#include<stdio.h>
#include<string.h>
int  main()
{
char dest[] = "Hello";
char src[] = " World";
strcat(dest, src);  // dest will become "Hello World"
printf("Concatenated string: %s", dest);
return 0;
}
```

Output: Concatenated string: Hello World

3. strcpy()

strcpy( ) function copies contents of one string into another string.

Syntax:

strcpy(destination string, source string);

Example:

```
#include<stdio.h>
#include<string.h>
int  main()
{
char dest[20];
char src[] = "Welcome";
strcpy(dest, src); // dest will become "Welcome"
printf("Copied string: %s", dest);
return 0;
}
```

Output: Copied String: Welcome

4.strcmp()

This function compares the strings and returns an integer value.

Syntax: strcmp(str1, str2);

It returns,

0 if the strings are equal
-1 if str1 is less than str2
1 if str1 is greater than str2

Example

```
#include <stdio.h>
#include <string.h>
int main() {
   char str1[] = "apple";
   char str2[] = "banana";
   int result = strcmp(str1, str2);
```

```
    if (result == 0) {
        printf("The strings are equal.\n");
    } else if (result < 0) {
        printf("String 1 is less than string 2.\n");
    } else {
        printf("String 1 is greater than string 2.\n");
    }
    return 0;
}
```

Output: String 1 is less than string 2.

5. strupr( ) :

The strupr( ) function is used to converts a given string to uppercase.

Syntax:

```
char *strupr(char *str);
```

Example:

```
int main()
{
    char str[ ] = "hello";
    //converting the given string into uppercase.
    printf("%s\n", strupr (str));
    return 0;
}
```

Output: HELLO

6.strlwr():

The strlwr( ) function is used to convert a given string into lowercase.

Syntax:

```
char *strlwr(char *str);
```

Example:

```
int main()
{
    char str[ ] = "HELLO";
    //converting the given string into lowercase.
    printf("%s\n", strlwr (str));
    return 0;
}
```

Output: hello

7. strrev():

The strrev() function is used to reverse the given string.

Syntax:

```
char *strrev(char *str);
```

Example:

```
#include <string.h>
int main()
```

```
{
 char str[50] = "Hello";
printf("The given string is =%s\n", str);
printf("After reversing string is =%s", strrev(str));
return 0;
}
```

Output:

```
The given string is = Hello
After reversing string is= olleH
```

c **Describe the following terms:** **4 M**

i) Keyword
ii) Identifier
iii) Variable
iv) Constant

Ans (Each term 1M)

(i)Keyword: Keywords are special words which have their own predefined meaning. The functions and meanings of these words cannot be altered. Some keywords are if, while, for, do, struct, switch etc.

(ii) Identifier: Identifiers are user-defined names of variables, functions and arrays. It comprises of combination of letters, digits and special symbol underscore (_).

Example

```
int age1;
float height_in_feet;
```

Here, *age1* is an identifier of integer data type.
Similarly *height_feet* is also an identifier but of float datatype,

(iii) Variable: a variable is a named memory location that can hold a data value. It acts as a placeholder for a value that can be changed during program execution. Variables are essential building blocks of any program, allowing you to store and manipulate data effectively.

Example: int age;

(iv) Constant:

Constants refer to fixed values that the program may not change during its execution. These fixed values are also called **literals.** Constants can be of any of the basic data types like an integer constant, a floating constant, a character constant, or a string literal.
Variable value may changes during its execution, but constant value is not changed during execution.

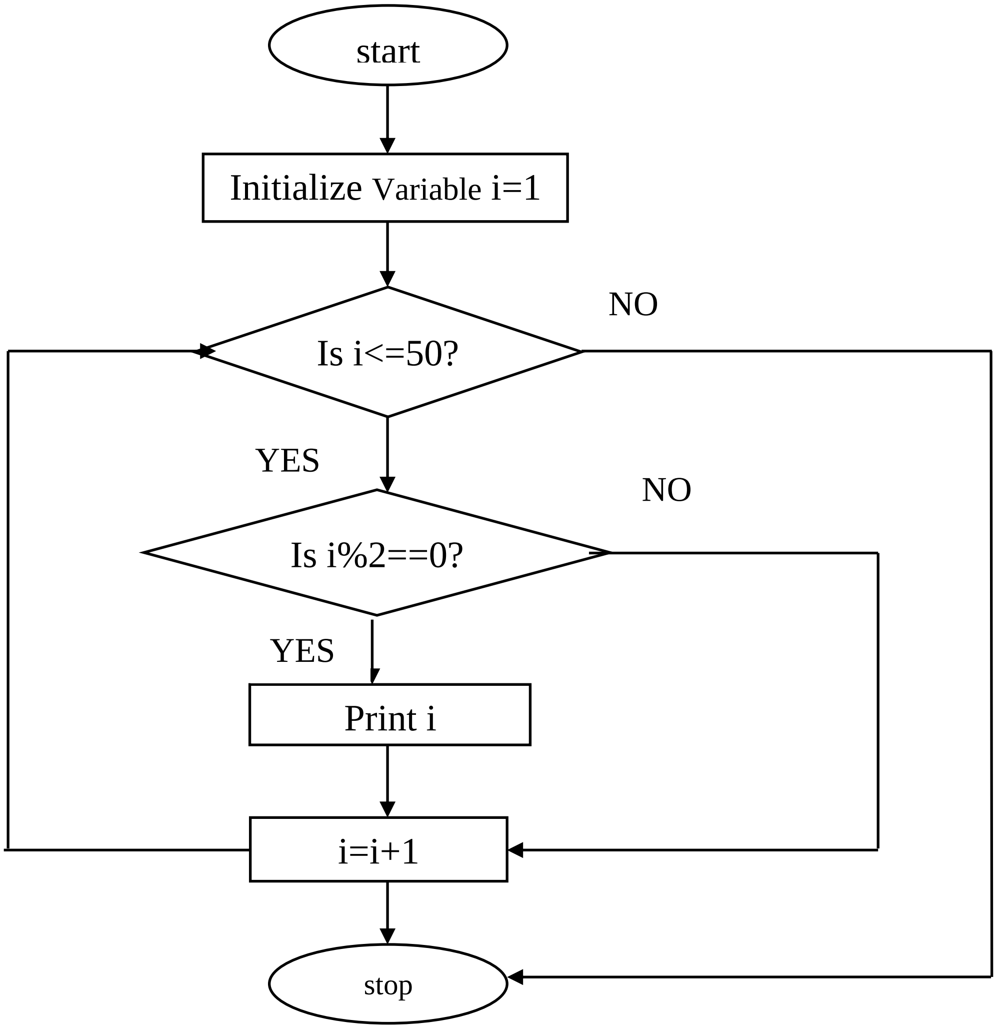
Example:

```
const int MAX_VALUE = 100; // Cannot be changed after initialization
const char GRADE = 'A';
```

d **Write a program to find area of circle using user defined function.** **4 M**

	Ans	**Note: Any type of function declaration and definition should be considered (with return value or no return value or with parameter or no parameter)**
		#include<stdio.h>
		#include<conio.h>
		void circle_area(float radius) //user defined function.
		{
		float area;
		area=3.14*radius*radius;
		printf("Area of circle= %f",area);
		}
		void main() //main function
		{
		float r;
		printf("Enter the radius of circle : ");
		scanf("%f", &r);
		circle_area(r);
		getch();
		}
			Main function 2M
			Function to calculate area 2M

4		**Attempt any THREE of the following:**	**12 M**
	a	**Write algorithm and draw flowchart to print even numbers from 1 to 50.**	**4 M**
	Ans	**Algorithm** 1. Start 2. Initialize the variable i to 1. 3. while i<=50 4. if i%2==0 5. print the number 6. increment value of i 7. stop **Flowchart**	Algorithm 2M Flowchart 2M

		 start Initialize Variable i=1 Is i<=50? NO YES Is i%2==0? NO YES Print i i=i+1 stop	
	b	**Explain increment and decrement operator with example.**	**4 M**

	Ans	**(Note: Any other correct example should be considered).**
		
		Increment operator
		Increment operator is used to increment or increase the value of a variable by one. It is equivalent to adding one to the value of the variable.
		The symbol used is ++.
		Syntax: ++var or var++ for increment
		Can be used in two ways:
		Prefix: ++variable (variable is incremented before its value is used).
		Postfix: variable++ (variable's original value is used first, then incremented).
		Example:
		Example:
		int i = 5;
		
		// Prefix increment
		i = ++i; // i becomes 6
			Explanation of each with example 2M

		int i=5;
		// Postfix increment
		int j = i++; // j becomes 5, i becomes 6
		Decrement operator
		The decrement operator is used to decrement or decrease the value of variable by 1. It is equivalent to subtracting one from the value of the variable.
		The symbol used is --.
		Syntax: --var or var-- for decrement.
		Can be used in two ways:
		Prefix: --variable (variable is decremented before its value is used).
		Postfix: variable-- (variable's original value is used first, then decremented).
		Example:
		int k = 7;
		
		// Prefix decrement
		k = --k; // k becomes 6
		
		int k=7;
		// Postfix decrement
		int l = k--; // l becomes 7, k becomes 6	

	c	**Write a program to sum all the odd numbers between 1 to 20.**	**4 M**

	Ans	**(Note: Any other correct logic should be considered).**
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int i,sum=0;
		clrscr();
		for(i=1;i<=20;i++)
		{
		if(i%2 !=0)
		{
		sum=sum+i;
		}
		}
		printf("Sum of all odd numbers between 1 to 20 =%d",sum);
		getch();
		}
			Finding odd numbers 2M
			
			Calculating sum 1M
			
			Display sum 1M

	d	**Illustrate initialization of one dimensional array with example.**	**4 M**
	Ans	**One dimensional array:** An array is a collection of variables of the same type that are referred through a common name. A specific element in an array is accessed by an index. all arrays consist of	Illustration 2M Example 2M

contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

Initialization:
Initialization can be done as compile time or runtime.

1. **Compile time:** This can be done by providing number of elements of the declared data type to an array at compile time.

```
 Eg :int arr[5]={1,2,3,4,5};
```

2. **Runtime:** For this loop structures like ‘for’ can be used to iterate through the locations of the array. Here the index of the array starts with 0 and ends with size minus one (size – 1) of an array.

```
 Eg :
int arr[5];
 for(i=0;i<5;i++)
 {
scanf(“%d”,&arr[i]);
}
```

e **Difference between call by value and call by reference.** **4 M**

Ans

Sr. No.	Call by value	Call by reference
1	When function is called by passing values then it is called as call by value	When function is called by passing address of variable then it is called as call by reference
2	Copy of actual variable is created when function is called.	No copy is created for actual variable rather address of actual variable is passed.
3	In call by value, memory required is more as copy of variable is created	In call by reference, memory required is less as there is no copy of actual variables.
4	Example : Function call – swap (x. y); Calling swap function by passing values	Example : Function call – swap (&x. &y); Calling swap function by passing address
5	Original (actual) parameters do not change. Changes take place on the copy of variable	Actual parameters change as function operates on value stored at the address
6	Address of the actual and formal arguments are different.	Address of the actual and formal arguments are the same.
7	Changes made inside the function is not reflected in other functions.	Changes made in the function is reflected outside function.

Any four differences 1M each

5		**Attempt any TWO of the following:**	**12 M**
	a	**Write a program to print fibonacci series starting from 0 and 1.**	**6 M**

	Ans	**(Note: Any other correct logic should be considered).**
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int a,b,c,limit,i;
		printf("\n Enter number:");
		scanf("%d",&limit);
		a=0;
		b=1;
		printf("%d \t %d",a,b);
		for(i=0;i<limit-2;i++)
		{
		c=a+b;
		printf("\t%d",c);
		a=b;
		b=c;
		}
		getch();
		}
			Correct logic with syntax 6M

	b	**Write a program for addition of two 3 x 3 matrices.**	**6 M**

	Ans	#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++)
		{
		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]);
			Declaration of variables 1M,
			Input matrices 2M,
			Calculating addition 2M,
			Display addition 1M

		}
		}
		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:\n");
		for(i=0;i<3;i++)
		{
		for(j=0;j<3;j++)
		{
		printf("%d ",c[i][j]);
		}
		printf("\n");
		}
		getch();
		}	

	c	**Write a program to compute the sum of all elements stored in an array using pointer.**	**6 M**

	Ans	(**Note: Any other correct logic shall be considered).**
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int a[5],sum=0,i,*ptr;
		clrscr();
		printf("\n Enter array elements:");
		for(i=0;i<5;i++)
		scanf("%d",&a[i]);
		ptr=&a[0];
		for(i=0;i<5;i++)
		{
		sum=sum+(*ptr);
		ptr=ptr+1;
		}
		printf("\n Sum= %d",sum);
		getch();
		}
			Variable declaration 1M,
			Input Array 1M,
			Pointer Initialization 1M,
			Sum calculation 2M,
			Display 1M

6		**Attempt any TWO of the following:**	**12 M**

	a	**Write a program to declare structure employee having data member name, age, city. Accept data for three employees and display it.**	**6 M**

	Ans	#include<stdio.h>
		#include<conio.h>
		struct employee
		{
		char name[10], city[10];
		int age;
		};
		void main()
		{
		int i;
		struct employee e[3];
		clrscr();
		for(i=0;i<3;i++)
		{
		printf("\n Enter name:");
		scanf("%s",&e[i].name);
		printf("\n Enter age:");
		scanf("%d",&e[i].age);
		printf("\n Enter city:");
		scanf("%s",&e[i].city);
		}
		for(i=0;i<3;i++)
		{
		printf("\n Name=%s",e[i].name);
		printf("\n Age=%d",e[i].age);
		printf("\n City=%s",e[i].city);
		}
		getch();
		}
			Declaration of Structure 2M
			Accepting data-2M
			Displaying Data 2M

	b	**Write a program to find factorial of a number using recursion.**	**6 M**

	Ans	#include<stdio.h>
		#include<conio.h>
		int factorial(int no)
		{
		if(no==1)
		return(1);
		else
		return(no*factorial(no-1));
		}
		void main()
		{
		int fact,no;
			Recursive function 4M,
			Main function 2M

		clrscr();
		printf("\n Enter number: ");
		scanf("%d",&no);
		fact=factorial(no);
		printf("\n Factorial number=%d",fact);
		getch();
		}	

	c	**Write a program to accept two numbers from user and perform addition, subtraction, multiplication and division operations using pointers.**	**6 M**

	Ans	#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int no1,no2,*ptr1,*ptr2,result;
		clrscr();
		printf("Enter no1:");
		scanf("%d",&no1);
		printf("\nEnter no2:");
		scanf("%d",&no2);
		ptr1=&no1;
		ptr2=&no2;
		result=(*ptr1)+(*ptr2);
		printf("\n Addition=%d",result);
		result=(*ptr1)-(*ptr2);
		printf("\n Subtraction=%d",result);
		result=(*ptr1)*(*ptr2);
		printf("\n Multiplication=%d",result);
		result=(*ptr1)/(*ptr2);
		printf("\n Division=%d",result);
		getch();
		}
			Accepting numbers 1M
			Pointer initialization-1M
			Addition 1M
			subtraction-1M
			multiplication-1M
			division-1M