V2V EDTECH LLP

Online Coaching at an Affordable Price.

OUR SERVICES:

- Diploma in All Branches, All Subjects
- Degree in All Branches, All Subjects
- BSCIT / CS
- Professional Courses

☎ +91 93260 50669

v2vedtech.com

▶ V2V EdTech LLP

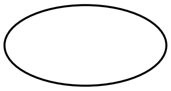
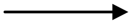
v2vedtech

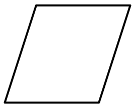
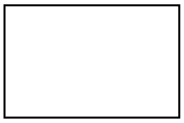
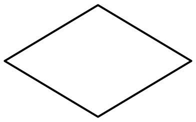
WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: 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.

<table>
<tr><th>Q. No.</th><th>Sub Q.N.</th><th>Answer</th><th>Marking Scheme</th></tr>
<tr><td><b>1.</b></td><td><b>(a)</b>
<b>Ans.</b></td><td><b>Attempt any FIVE of the following:</b>
<b>Define array. List its type.</b>
<b>Array</b> is a fixed-size sequential collection of elements of the same type.
<b>Types:</b>
1. One dimensional
2. Multi dimensional</td><td><b>10</b>
<b>2M</b>
<b><i>Definition 1M</i></b>
<b><i>Types 1M</i></b></td></tr>
<tr><td></td><td><b>(b)</b>
<b>Ans.</b></td><td><b>Draw &amp; label different symbols used in flowcharts.</b>
<table>
<tr><th>Symbol</th><th>Name</th><th>Function</th></tr>
<tr><td></td><td>Start/end</td><td>An oval represents a start or end point</td></tr>
<tr><td></td><td>Arrows</td><td>A line is a connector that shows relationships between the representative shapes</td></tr>
</table>
</td><td><b>2M</b>
<b><i>Any 4 symbols ½M each</i></b></td></tr>
</table>

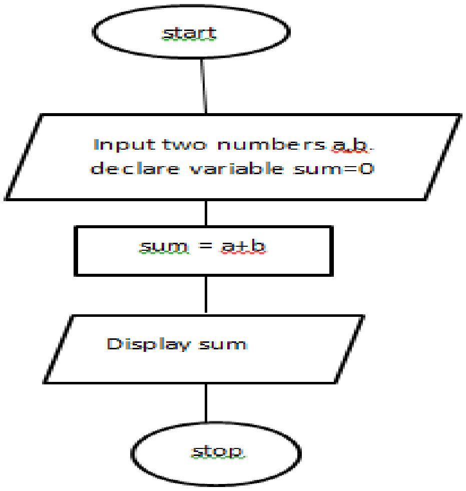
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

	(c)
	
	Ans.
		Find the output of the following program:
		#include<stdio.h>
		void main( )
		{
		int x = 10, y = 10, v1, v2;
		v1 = x++;
		v2 = ++y;
		printf(“value of v1: %d, v1);
		printf(“value of v2: %d, v2);
		}
		
		Output:
		value of v1:10value of v2:11
			2M
			
			Correct output 2M

	(d) Ans.	**State the syntax & use of strlen ( ) & strcat ( ) function.** **strlen( )**: calculates the length of the string *Syntax:* strlen(s1); **strcat():**concatenates two strings *Syntax:* strcat(s1,s2)	**2M** ***1M for correct syntax*** ***1M for use***
	(e) Ans.	**State the Relational operators with example.** == - returns true if the values of two operands are equal else returns false. E.g: if (A= = B){ } != - returns true if values of two operands are not equal, else returns	**2M**

		false
		E.g: if (A! = B){ }
		<- returns true if the first operand is less than the second, else returns false.
		E.g: if (A< B){ }
		>- returns true if the first operand is greater than the second, else returns false.
		E.g: if (A> B){ }
		<= returns true if the first operand is less than or equal to the second, else returns false.
		E.g: if (A< = B){ }
		>= returns true if the first operand is greater than or equal to the second, else returns false.
		E.g: if (A> = B){ }
			Any four operators ½M each

	**(f)
	Ans.**
		State the syntax to declare pointer variable with example.
		General syntax to declare pointer.
		datatype *var_name;
		Eg: int var = 20;
			2M
			Correct syntax 1M
			Correct example 1M

	(g) Ans.	**Draw flow chart for addition of two numbers.**  start Input two numbers a,b. declare variable sum=0 sum = a+b Display sum stop	**2M** ***Correct sequence 1M*** ***Correct symbol 1M***

2. **Attempt any THREE of the following:** **12**

(a) **State the importance of flow chart.** **4M**

Ans. A flowchart is a type of diagram that represents an algorithm. It is a visual representation of a sequence of steps to complete the process. A flow chart describes a process using symbols rather than words. Computer programmers use flow charts to show where data enters the program, what processes the data goes through, and how the data is converted to output.

-can be used to quickly communicate the ideas or plans that one programmer envisions to other people who will be involved in the process.

- aid in the analysis of the process to make sure nothing is left out and that all possible inputs, processes, and outputs have been accounted for.

-help programmers develop the most efficient coding because they can clearly see where the data is going to end up.

- help programmers figure out where a potential problem area is and helps them with debugging or cleaning up code that is not working.

- are a useful tool in visualizing a module's flow of execution before writing any code. This allows developers to do three things: verify the algorithm's correctness before writing code, visualize how the code will ultimately be written, and communicate and document the algorithm with other developers and even non-developers.

-may be used in conjunction with other tools, such as pseudo-code, or may be used by itself to communicate a module's ultimate design, depending on the level of detail of the flowchart.

Any 4 points 1M each

(b) **Write a program to declare structure student having rollno, name & marks.** **4M**

***(Note: Any other correct logic shall be considered)*.**

Ans. Accept and display data for three students.

```
#include<stdio.h>
#include<conio.h>
void main() {
int i;
struct student{
int rollno;
char name[20];
int marks;
} s[3];
```

Correct logic 3M

Correct syntax 1M

```
clrscr();
for(i=0;i<3;i++) {
printf("Enter rollno, name and marks\n");
scanf("%d%s%d",&s[i].rollno,&s[i].name,&s[i].marks);
}
for(i = 0; i<3;i++){
printf("\nThe details of student %d\n",i+1);
printf("Roll no %d\n",s[i].rollno);
printf("Name is %s\n",s[i].name);
printf("Marks %d\n",s[i].marks);
}
getch();
}
```

(c) **Explain pointer arithmetic with example.** **4M**

(Note: Code snippet shall be considered).

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

Any two operators

Each operator with explanation 1M

1M for each example

Decrement

It is used to decrement the pointer. Each time a pointer is decremented, it points to the previous location. Eg, if the current position of pointer is 1002, then decrement operation results in the pointer pointing to the location 1000.

Addition and subtraction:

When addition or subtraction operation is performed on the pointer variable, it shows that particular location in the memory.

Eg: int *ptr; -say address is 1000.

If -> ptr+n- then ptr+n*2 .

If -> ptr-n thenptr-n*2.

```
#include<stdio.h>
#include<conio.h>
void main() {
 int i = 10;
 int *ptr=&i;
 clrscr();
 printf("%x%d",ptr,i);
 ptr++;
 printf("\n%x%d",ptr,i);
 printf("\n%x",ptr+2);
 printf("\n%x",ptr-2);
getch();
}
```

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

(Note: Any example shall be considered)

Ans. When a series of decision is required, nested if-else is used. Nesting means using one if-else construct within another one. If the condition in the outer if, is true, then only the inner if-else will get executed. Further the statements in the inner if will get execute only if the condition of inner if, evaluates to true. If it is false, the statements in inner else will get executed.

If the outer if evaluates to false, then the statements in outer else get executed.

Explanation 2M

General syntax:

```
if(condition) {
        if(condition) {
                statements
        } else {
                statements
        }
} else {
        statements
}
statements
```

Example:

```
#include<stdio.h>
#include<conio.h>
void main() {
int val;
clrscr();
```

Example 2M

```
printf("Enter a number");
scanf("%d",&val);
if(val>=5) {
if(val>5) {
printf("Number is greater than 5");
} else {
  printf("Number is equal to 5");
}
  } else {
 printf("Number is less than 5");
 }
 getch();
}
```

3.		**Attempt any THREE of the following:**	12
	(a)	**Describe the following terms:** **(i) Keyword** **(ii) Identifier** **(iii) Variable** **(iv) Constant**	4M
	Ans.	**(i) Keyword:** Keywords are special words in C programming which have their own predefined meaning. The functions and meanings of these words cannot be altered. Some keywords in C Programming are if, while, for, do, etc.. **(ii) Identifier:** Identifiers are user-defined names of variables, functions and arrays. It comprises of combination of letters and digits. *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 floating integer data type, **(iii) Variable:** A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in C has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable. *Example*: add, a, name **(iv) Constant:**	***Each term 1M***

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. There are enumeration constants as well.

Example:
121
234
3.14

(b) Ans. **Differentiate between call by value and call by reference.** **4M**

Sr. No.	Call by value	Call by reference
1	When function is called by passing values then it is 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 generated 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.

Any four differences 1M each

(c) Ans. **Explain conditional operator with example.** **4M**

Conditional Operator (Ternary Operator):

It takes the form „? :‟ to construct conditional expressions

The operator „? :‟ works as follows:

exp1 ? exp2 : exp 3

Where exp1, exp2 and exp3 are expressions.exp1 is evaluated first, If it is true, then the expression exp2 is evaluated and becomes the value

Explanation 2M

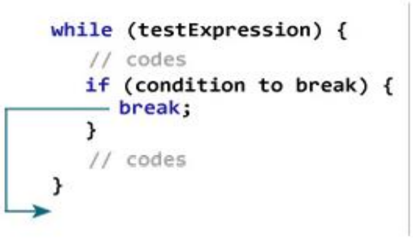
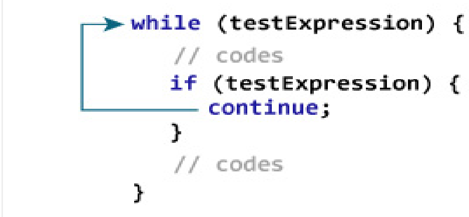
Example

		of the expression. If exp1 is false, exp3 is evaluated and its value becomes the value of the expression.
		E.g. int a=10,b=5,x;
		x=(a>b) ? a : b;
			2M

	(d)
	Ans.
		List the categories of functions and explain any one with example.
		Different categories of function:
		1) Function with no arguments and no return value.
		2) Function with arguments and no return value.
		3) Function with no arguments and return value.
		4) Function with arguments and return value.
		1) Function with no arguments and no return value:
		This category of function cannot return any value back to the calling program and it does not accept any arguments also. It has to be declared as void.
		For example:
		void add()
		{
		inta,b,c;
		a=5;
		b=6;
		c=a+b;
		printf(“%d”,c);
		}
		It should be called as add();
		2) Function with arguments and no return value:
		This category of function cannot return any value back to the calling program but it takes arguments from calling program. It has to be declared as void. The number of arguments should match in sequence, number and data type.
		For example:
		void add(intx,int y)
		{
		int z;
		z=x+y;
		printf(“%d”,z);
		}
		It should be called as add(4,5); where x will take 4 and y will take 5 as their values.
			4M
			List 2M
			Explanation of any one category 2M

		3) Function with no arguments and return value:
		This category of function can return a value back to the calling program but it does not take arguments from calling program. It has to be declared with same data type as the data type of return variable.
		For example:
		int add()
		{
		inta,b,c;
		a=5;
		b=6;
		c=a+b;
		return(c);
		}
		It should be called as int x = add(); where x will store value returned by the function.
		4) Function with arguments and return value:
		This category of function can return a value back to the calling program but it also takes arguments from calling program. It has to be declared with same data type as the data type of return variable.
		For example:
		int add(intx,int y)
		{
		int z;
		z=x+y;
		return(z);
		}
		It should be called as int s = add(4,5); where x will have 4 and y will have 5 as their values and s will store value returned by the function.	

4.		**Attempt any THREE of the following:**	**12**
	(a)	**Write an algorithm to determine the given number is odd or even.**	**4M**
	Ans.	Step 1- Start Step 2- Read / input the number. Step 3- if n%2==0 then number is even. Step 4- else number is odd. Step 5- display the output. Step 6- Stop	***Correct algorith m 4M***
	(b)	**Illustrate the use of break and continue statement with example.**	**4M**

	Ans.	***(Note:- Any other example shall be considered)***
		Break: It breaks the execution of the loop which allows exiting from any loop or switch, such that break statement skips the remaining part of current iterations of the loop.
		Syntax: break;
		
		while (testExpression) {
		    // codes
		    if (condition to break) {
		        break;
		    }
		    // codes
		}
		Continue: It is used when it is required to skip the remaining portion of the loop without breaking loop it will transfer control directly to next iteration
		Syntax: continue;
		
		while (testExpression) {
		    // codes
		    if (testExpression) {
		        continue;
		    }
		    // codes
		}
		In given program sequence if “break” executes then execution control will jump out of loop & next statement after loop will be executed. In given program sequence if “continue” executes then execution control will skip remaining statements of loop & will start next iteration of loop
			Use of each 1M
			Example of each 1M

	(c)
	Ans.
		Write a program to add, subtract, multiply and divide two numbers, accepted from user switch case.
		(Note: Any other correct logic shall be considered).
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int a,b,ch,add,sub,mul,div;
		clrscr();
		printf("\n1 for addition \n2 for substraction");
		printf("\n3 for multiplication \n4 for division");
		printf("\nEnter two numbers:");
			4M
			Correct logic 2M
			Correct syntax 2M

		scanf("%d%d",&a,&b);
		printf("\nEnter your choice:");
		scanf("%d",&ch);
		switch(ch)
		{
		case 1:
		add=a+b;
		printf("Addition of a & b=%d",add);
		break;
		case 2:
		sub=a-b;
		printf("Substraction of a & b=%d",sub);
		break;
		case 3:
		mul=a*b;
		printf("Multiplication of two numbers=%d",mul);
		break;
		case 4:
		div=a/b;
		printf("Division of two numbers=%d",div);
		break;
		default:
		printf("Invalid choice....");
		}
		getch();
		}	

	(d) **Ans.**	**Illustrate initialization of two dimensional array with example.** Two dimensional array: The array which is used to represent and store data in a tabular form is called as two dimensional array. Such type of array is specially used to represent data in a matrix form. Initialization can be done as design time or runtime. 1. Design time: This can be done by providing „row X column‟ number of elements to the array. Eg for a 3 rows and 4 columns array , 3X4=12 elements can be provided as : arr[3][4]={ {2,3,4,6}, {1,4,6,3}, {6,6,4,3}, {6,7,8,9} };	**4M** ***Two dim array 1M*** ***Declaration 1M***

		2. Runtime: For this loop structures like „for‟ can be used in a nested form, where outer loop will increment row and inner loop will increment column.
		Eg :
		for(i=0;i<3;i++)
		{
		for(j=0;j<4;j++)
		{
		scanf("%d", &arr[i][j]);
		}
		}
		Example:
		main()
		{
		int arr[2][2]={{1,2},{4,5}};
		int i,j;
		for(i=0;i<2;i++)
		{
		for(j=0;j<2;j++)
		{
		printf("%d", arr[i][j]);
		}
		printf("\n");
		}
		}
			Initialization by any one type 1M
			Example 1M

	(e)
	Ans.
		Write a program to read two strings and find whether they are equal or not.
		(Note: Any other correct logic shall be considered).
		#include<stdio.h>
		#include<conio.h>
		#include<string.h>
		void main()
		{
		char st1[20],st2[20];
		printf("enter string 1");
		scanf("%s",st1);
		printf("enter second string");
		scanf("%s",st2);
		if(strcmp(st1,st2)==0)
			4M
			Correct logic 2M
			Correct syntax 2M

		printf(“\nboth strings are equal”);
		else
		printf(“\nstrings are not equal”);
		}	

5.	**(a)**
	
	Ans.
		Attempt any TWO of the following:
		Write a program to calculate sum of all the odd numbers between 1 to 20.
		(Note: Any other correct logic shall be considered).
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		inti,sum=0;
		clrscr();
		for(i=1;i<=20;i++)
		{
		if(i%2!=0)
		{
		sum=sum+i;
		}
		}
		printf("Sum=%d",sum);
		getch();
		}
			12
			6M
			
			Finding odd numbers 2M
			
			Calculating sum 1M
			
			Display sum 1M
			
			Correct syntax 2M

	(b)
	
	Ans.
		Write a program for addition of two 3 x 3 matrices.
		(Note: Any other correct 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("\n Enter first matrix");
		for(i=0;i<3;i++)
		{
		for(j=0;j<3;j++)
		{
		scanf("%d",&a[i][j]);
		}
			6M
			
			Deceleration of variables 1M
			
			Input matrices 2M

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

Calculating addition 2M

Display addition 1M

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

(Note: Any other correct logic shall be considered).

Ans.

```
#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]);
```

Variable declaration 1M

Input array 1M

		ptr=&a[0];
		for(i=0;i<5;i++)
		{
		sum=sum+(*ptr);
		ptr=ptr+1;
		}
		printf("\n Sum= %d",sum);
		getch();
		}
			Pointer Initialization 1M
			Sum calculation 2M
			Display 1M

6.	**(a)**
	Ans.
		Attempt any TWO of the following:
		Write a program to sort elements of an array in ascending order.
		(Note: Any other correct logic shall be considered).
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int a[5],i,j,temp;
		clrscr();
		printf("\n Enter array elements:");
		for(i=0;i<5;i++)
		scanf("%d",&a[i]);
		for(i=0;i<5;i++)
		{
		for(j=0;j<4-i;j++)
		{
		if(a[j]>a[j+1])
		{
		temp=a[j];
		a[j]=a[j+1];
		a[j+1]=temp;
		}
		}
		}
		for(i=0;i<5;i++)
		printf("\n %d",a[i]);
		getch();
		}
			12
			6M
			Input array 1M
			Sorting logic 4M
			Display sorted list 1M

	(b)	**Write a function to print Fibonacci series starting from 0, 1.** ***(Note: Any other correct logic shall be considered).***	**6M**

	Ans.	void Fibbo()
		{
		inta,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;
		}
		}
			Correct function with syntax 6M

	(c)
	
	Ans.
		Calculate factorial of a number using recursion.
		(Note: Explanation/algorithm/program shall be considered)
		#include<stdio.h>
		#include<conio.h>
		int factorial(int no)
		{
		if(no==1)
		return(1);
		else
		return(no*factorial(no-1));
		}
		void main()
		{
		intfact,no;
		clrscr();
		printf("\n Enter number");
		scanf("%d",&no);
		fact=factorial(no);
		printf("\n Factorial number=%d",fact);
		getch();
		}
			6M
			
			Recursiv e function 4M
			
			Main function 2M