1. Differentiate between array and structure with suitable examples.

	Arrays	**Structures**
1.	An array is a collection that consists of elements of homogeneous data type i.e. same data type	A Structure is a collection that consists of elements of heterogeneous data type i.e. different data types.
2.	Array is declared using '[ ]'.	Declared using the keyword 'struct'.
3.	Array uses subscripts or '[ ]' (square brackets) to access the elements	Structure uses the '.' (dot operator) to access the elements.
4.	The size of an array is fixed	The size of a structure is not fixed.
5.	Traversing through and searching for elements in an array is quick and easy.	Traversing through and searching for elements in a structure is slow and complex.
6.	An array is always stored in contiguous memory locations.	A structure may or may not be stored in contiguous memory locations.
7.	Array elements are accessed by their index number using subscripts.	Structure elements are accessed by their names using dot operator.
8.	Array is pointer as it points to the first element of the collection.	Structure is not a pointer.

9.	Ex: int marks [5] = {20, 30, 40, 50, 60};	struct student
		{
		char name;
		int mark;
		float fees;
		}

2. What is a pointer? How the pointer variables are declared?

A pointer is a variable that contains the memory location of another variable. A pointer is variable that represents the location of a data item, such as a variable or an array element.

Declaring pointer variables: The syntax for declaring pointer variable is given as:

```
data_type   *ptr_name;
```

Here, data_type is the data type of the value that the pointer will point.

For example:

```
int *num;
char *ch;
float *avg;
```

In above examples, pointer variables are declared to point to a variable of the specified data type.

3. Define structure. How structure variables are declared & initialized? Give examples.

A structure is a user defined data type that can store related information (even of different data types) together. It is similar to records and can be used to store information about an entity.

Structure Declaration

A structure is declared using the keyword 'struct' followed by the structure name. All the variables of the structure are declared within the structure. A structure type is generally declared by using the following syntax:

```
struct struct-name
{
data_type   var-name;
data_type   var-name;
............................
};
```

Example: To define a structure for a student, then the related information would be: roll_number, name, course, and fees. This structure can be declared as:

```
struct student
{
int roll_no;
char name[20];
char course[20];
float fees;
}
```

Initialization of Structures

Initializing a structure means assigning some constants to the members of the structure. A structure can be initialized in the same way as other data types are initialized.

The initializers are enclosed in braces and are separated by commas. The general syntax to initialize a structure variable is given as follows:

```
struct struct_name
{
data_type    member_name1;
data_type    member_name2;
data_type    member_name3;
....................................
} struct_var {constant1, constant2, = constant 3, ........ };
```

Example: We can initialize s student structure by writing:

```
struct student
{
```

```
int roll_no;
char name[20];
char course[20];
float fees;
} stud1 = { 01, “RAJA”, “MCA”, 45000 };
```

4. Explain with suitable examples the how the members of a structure are accessed.

A structure member variable is generally accessed using a **.** (dot) operator. The syntax of accessing a structure or a member of a structure can be given as follows:

```
struct_var . member_name
```

The dot operator is used to select a particular member of the structure.

For example, to assign value to the individual data members of the structure variable stud1, we may write

```
stud1.roll_no = 01;
stud1.name = "Rahul";
stud1.course = "BCA";
stud1.fees = 45000;
```

To input values for data members of the structure variable stud1, we may write

```
scanf ( "%d", &stud1.r_no );
scanf ( "%s", stud1.name );
```

Similarly, to print the values of structure variable stud1, we may write

```
printf ( "%s", stud1.course );
printf ( "%f", stud1.fees );
```

Memory is allocated only when we declare variables of the structure.

Once the variables of a structure are defined, we can perform a few operations on them. For example, we can use the assignment operator '=' to assign the values of one variable to another.

5. List all string handling functions. Explain any four with syntax and examples.

The different string handling functions are:

strcat, strncat, strchr, strrchr, strcmp, strncmp, strcpy, strncpy, strle, strstr, strspn, strcspn.

strcat Function	**strncat Function**
Syntax: char *strcat (char *str1, const char *str2);	Syntax: char *strncat (char *str1, const char *str2, size_n);
The strcat function appends the string pointed to by str2 to the end of the string pointed to by str1. The terminating null character of str1 is overwritten. The process	The strncat function appends the string pointed to by str2 to the end of the string pointed to by str1 up to n characters long. The terminating null character of str1 is

stops when the terminating null character of str2 is copied. The argument str1 is returned.	overwritten. Copying stops when n characters are copied or the terminating null character of str2 is copied.

#include <stdio.h>
#include <string.h>
int main ( )
{
char str1[50] = "Programming";
char str2[ ] = "In C";
strcat (str1, str2);
printf ("Resulting string is: %s", str1);
return 0;
}
Output:
Resulting string is: Programming In C
	#include <stdio.h>
	#include <string.h>
	int main ( )
	{
	char str1[50] = "Programming";
	char str2[ ] = "In C";
	strncat (str1, str2, 2);
	printf ("Resulting string is: %s", str1);
	return 0;
	}
	Output:
	Resulting string is: Programming In

strcmp Function	**strncmp Function**
Syntax: int strcmp (const char *str1, const char *str2);	Syntax: int strncmp (const char *str1, const char *str2, size_n);
This function compares the string pointed by str1 to the string pointed by str2. The function returns zero if the strings are equal. Otherwise, it returns a value less than zero or greater than zero if str1 is less than or greater than str2, respectively.	This function compares at most the first n bytes of str1 and str2. The process stops comparing after the null character is encountered. The function returns zero if the first n bytes of strings are equal. Otherwise, it returns a value less than zero or greater than zero if str1 is less than or greater than str2, respectively.

#include <stdio.h>
#include <string.h>
int main()
{
char str1[10]= "HELLO";
char str2[10]= "HEY";
if (strcmp (str1, str2) ==0)
printf ("Two strings are identical");
else
printf ("Two strings are not identical");
	#include <stdio.h>
	#include <string.h>
	int main()
	{
	char str1[10]= "HELLO";
	char str2[10]= "HEY";
	if (strncmp (str1, str2, 2) ==0)
	printf ("Two strings are identical");
	else
	printf ("Two strings are not identical");

return 0;
}
Output:
Two strings are not identical
	return 0;
	}
	Output:
	Two strings are identical

strstr Function	**strlen Function**
Syntax: char *strstr (const char *str1, const char *str2);	Syntax: size_t strlen (const char *str);
This function is used to find the first occurrence of string str2 in the string str1. It returns a pointer to the first occurrence of str2 in str1. If no match is found, then a null pointer is returned.	This function calculates the length of the string str up to but not including the null character, i.e., the function returns the number of characters in the string.

#include <stdio.h>
#include <string.h>
int main ( )
{
char str1[ ] = "HAPPY BIRTHDAY TO YOU";
char str2[ ] = "DAY";
char *ptr;
ptr = strstr (str1, str2);
if (ptr)
printf ("Substring Found");
else
printf ("Substring Not Found");
return 0;
}
Output: Substring found
	#include <stdio.h>
	#include <string.h>
	int main ( )
	{
	char str[ ] = "HELLO";
	printf ("Length of str is: %d", strlen(str));
	return 0;
	}
	Output: Length of str is: 5

strcpy Function	**strncpy Function**
Syntax: char *strcpy (char *str1, const char *str2);	Syntax: char *strncpy (char *str1, const char *str2, size_n);
This function copies the string pointed to by str2 to str1 including null character of str2. It returns the argument str1. Here str1 should be big	This function copies up to n characters from the string pointed to by str2 to str1. Copying stops when n characters are copied. If null character of str2 is reached then null character is continually

enough to store the contents of str2.	copied to str1 until n characters are copied.
#include <stdio.h>	#include <stdio.h>
#include <string.h>	#include <string.h>
int main()	int main()
{	{
char str1[10], str2[10]= "HELLO";	char str1[10], char str2[10]= "HELLO";
strcpy (str1, str2);	strncpy (str1, str2, 3);
printf (" Resulting string is: %s", str1);	printf (" Resulting string is: %s", str1);
return 0;	return 0;
}	}
Output:	Output:
Resulting string is: HELLO	Resulting string is: HEL

6. Write a C program to add/multiply two numbers using pointers.

#include <stdio.h>	#include <stdio.h>
int main( )	int main( )
{	{
int n1, n2, *ptr1, *ptr2, sum;	int n1, n2, *ptr1, *ptr2, mul;
printf ("Enter two numbers: \n");	printf ("Enter two numbers: \n");
scanf ("%d %d", &n1, &n2);	scanf ("%d %d", &n1, &n2);
ptr1 = &n1;	ptr1 = &n1;
ptr2 = &n2;	ptr2 = &n2;
sum = *ptr1 + *ptr2;	mul = *ptr1 * *ptr2;
printf ("The result is = %d", sum);	printf ("The result is = %d", mul);
return 0;	return 0;
}	}

7. Write a C program to swap two integer values using pointers.

```
#include <stdio.h>
int main ( )
{
int x, y, *a, *b, temp;
printf ( "Enter the value of x and y \n" );
scanf ( "%d %d", &x, &y );
printf ( "Before Swapping: x = %d  \n  y = %d", x, y);
a = &x;
b = &y;
temp = *b;
b = *a;
*a = temp;
printf ( "After Swapping: x = %d   \n  y = %d\n", x, y);
return 0;
}
```

8. Develop a C program to concatenate two strings without using built-in function.

```
#include <stdio.h>
int main ( )
{
char str1[100], str2[100], str3[100];
int i = 0, j = 0;
printf ( "Enter the first string: " );
gets ( str1 );
printf ( "Enter the second string: " );
gets ( str2 );
while (str1[1] != '\0' )
{
str3[ j ] = str1[ i ];
i++;
j++;
}
i=0;
while ( str2[ i ] != '\0 ')
{
str3[ j ] = str2[ i ];
i++;
j++;
}
str3[ j ] = '\0';
printf ( "The concatenated string is: " );
puts( str3 );
return 0;
}
```

9. Write a program to append (copy) a string to another string without using built-in function.

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

```
{
char dest[100], source[50];
int i = 0, j = 0;
printf ( "Enter the source string: " );
gets ( source );
printf ( "Enter the destination string: " );
gets ( dest );
while ( dest[i] != '\0' )
i++;
while ( source[j] != '\0' )
{
dest[ i ] = source[ j ];
i++;
j++;
}
dest[ i ] = '\0';
printf ( "After appending, the destination string is: " );
puts( dest );
return 0;
}
```

10. Write a program to reverse the given string.

```
#include <stdio.h>
int main ( )
{
{
char str[100], temp;
int i = 0, j = 0;
printf ( "Enter the string: " );
gets ( source );
j = strlen(str) – 1;
while ( i < j )
{
temp = str[ j ];
str[ j ] = str[ i ];
str[ i ] = temp;
i++;
j--;
}
printf ( "The reversed string is: " );
puts( str );
return 0;
}
```

11. Write a C program to read a sentence & count the number of words in the sentence.

```
#include <stdio.h>
int main ( )
{
char str[200];
int i = 0, count = 0;
printf ( "Enter the sentence: ");
gets( str );
while ( str[ i ] != '\0')
{
if ( str[ i ] == ‘   ‘   &&   str[ i+1 ] != ‘   ‘ )
count++;
i++;
}
printf ( "The total count of words is: %d", count+1);
return 0;
}
```

12. Write a C program to implement structure to read, write and compute average marks and the students scoring above and below the average marks for a class of N students.

```
#include<stdio.h>
        struct student
        {
        char usn[10];
        char name[10];
        float m1, m2, m3;
        float avg, total;
        };
void main( )
{
struct student s[20];
int n, i;
printf ( "Enter the number of students:" );
scanf ( "%d", &n );
for ( i=0; i<n; i++ )
{
printf ( "Enter the detail of %d student: \n",i+1 );
printf ( "Enter USN: \n" );
scanf ( "%s", s[i].usn );
printf ( "Enter Name: \n ");
scanf ( "%s", s[i].name );
printf ( "Enter the three subject score: \n" );
scanf ( "%f %f %f", &s[i].m1, &s[i].m2, &s[i].m3 );
s[i].total = s[i].m1 + s[i].m2 + s[i].m3;
s[i].avg = s[i].total/3;
}
printf ( "\n Student details are: \n" );
printf ( "USN \t\t Name \t\t Marks \t\t Average\n" );
printf( "------------------------------------------------------------- \n");
for (i=0; i<n; i++ )
printf ( "%s \t %s \t\t %.2f %.2f %.2f \t%.4f\n", s[i].usn, s[i].name, s[i].m1 ,s[i].m2, s[i].m3,
s[i].avg);
for ( i=0; i<n; i++ )
{
if ( s[i].avg >= 35 )
printf ( "\n %s has scored above the average marks", s[i].name );
else
printf ( "\n %s has scored below the average marks", s[i].name );
}
}
```

13. Develop a program using pointers to compute the Sum, Mean and Standard deviation of all elements stored in an array of N real numbers.

```
#include<stdio.h>
#include<math.h>
void main( )
{
int n , i;
float x[20], sum, mean;
float variance, deviation;
printf ("Enter the value of n: \n" );
scanf ( "%d", &n );
printf ( "Enter the values: \n" );
for ( i=0; i<n; i++ )
{
scanf ( "%f",(x+i) );
}
sum = 0;
for ( i=0; i<n; i++ )
{
sum = sum + *(x+i);
}
printf ( "sum = %f\n", sum );
mean = sum/n;
sum = 0;
for ( i=0; i<n; i++ )
{
sum = sum + (*(x+i)-mean) * (*(x+i)-mean);
}
variance = sum/n;
deviation = sqrt( variance );
printf ( "mean(Average) = %f \n", mean);
printf ( "variance = %f \n", variance );
printf ( "standard deviation = %f\n", deviation );
}
```

14. Write a program to read and print the names of n students of a class.

```
#include <stdio.h>
int main ( )
{
char names[5][15];
int i, n;
 printf ("Enter the number of students: \n" );
scanf ( "%d", &n );
for ( i=0; i<n; i++ )
{
printf ( “Enter the name of student %d:, i+1);
gets ( names[i] );
}
printf ( “The names of students are: \n”);
for ( i=0; i<n; i++ )
puts ( names[i] );
return 0;
}
```

15. Write a program to convert characters of s string into lowercase / uppercase.

Covert string to Lowercase	**Covert string to Uppercase**
#include <stdio.h>	#include <stdio.h>
int main ( )	int main ( )
{	{
char str[100], lowercase[100];	char str[100], uppercase[100];
int i = 0, j = 0;	int i = 0, j = 0;
printf ("Enter the string: ");	printf ("Enter the string: ");
gets (str);	gets (str);
while (str[i] != '\0')	while (str[i] != '\0')
{	{
if (str[i] >= ‘ A ‘ && str[i+1] <= ‘ Z ‘)	if (str[i] >= ‘ a ‘ && str[i+1] <= ‘ z ‘)
lowercase[j] = str[i] + 32;	uppercase[j] = str[i] - 32;
else	else
lowercase[j] = str[i];	uppercase[j] = str[i];
i++; j++;	i++; j++;
}	}
lowercase[j] = ‘ \0 ’ ;	uppercase[j] = ‘ \0 ’ ;
printf { “Lowercase string is : \n”);	printf { “Uppercase string is : \n”);
puts(lowercase);	puts(uppercase);
return 0;	return 0;
}	}

16. Define a structure to store: Information of book, Customer information.

Customer Information	**Book Information**
struct customer	struct book
{	{
int cust_id;	char title[30];
char name[12];	char author[25];
char address[25];	int pages;
int mobile_no;	float price;
int DOB;	int publication;
}	}