1) Write a C program that defines a structure employee containing the details such as employee Number, employee Name, department name and salary. The structure has to store 20 employee in an organization

A

```
# include < stdio.h >
# include < string.h >
Struct employee
{
int employee ;
char emp Name [50];
Char dept Name [50];
float salary ;
};
int main ( )
{
Struct employee employee [20];
int i ;
printf (" enter details for 20 employees : \n");
for (i=0 ; i< 20; i++ )
```

```
{
printf ("\n employee : %d : \n", i+1);
printf (" enter employee number : ");
scanf ("%d", &employee[i].empnumber);
printf (" enter employee name : ");
scanf ("%s", employee[i].empName);
printf (" enter department name : ");
scanf ("%s", employee[i].deptName);
printf (" enter Salary : ");
Scanf ("%b", &employee.[i].Salary);
}
printf ("\n employee : Deatails : \n");
for (i=0; i<20; i++)
{
printf ("\n employee %d : \n", i+1);
printf (" employee Number : %d \n", employee[i].empNumber);
printf (" employee Name : %s \n", employee[i].empName);
printf (" Department Name : %s \n", employee[i].deptName);
printf (" Salary : %2f \n", employee[i].salary);
}
return 0;
}
```

6/12

OUTPUT

Enter employee Id : 5

Enter Name : Ali

Enter Salary : 45000

Id : 05

Name : Ali

Salary : 45000.00

2 list out the difference between Unions, structures and Arrays

Union	Structures	Arrays
• Groups variables of different data types, sharing the same memory space	• Groups variables of different datatypes with separate memory spaces	• Stores multiple elements of the same data type contigous memory
equal to the size of the largest member	• Memory allocation is equal to sum of all member sizes	• equal to the size of the element type a number of elements
Only one member holds a valid value at a time	All members can hold valid values simultaneously	Accessed using indices all elements hold valid values
Can store different data types but one at a time	Can store different data types simultaneously	• Only stores elements of the same data type

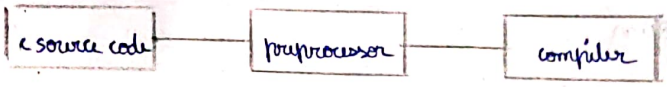
Slightly faster than structures because only one member is stored at a time	Access time depends on individual members all are stored in separate memory location	Access time is constant for each element (due to index addressing)
Used when memory optimization is critical eg:- in embedded systems or hardware programming	Used for logically grouping related data eg:- student records	Used for storing and manipulating collection of similar items eg: lists or matrices
Syntax: Union Unionname { Data type member 1; Datatype member 2; ... Data type member n; }	Syntax struct struct name { data type member name 1; ... } variable 1, variable 2 ...);	Syntax .1-D Datatype arrayname [array size] .2-D Datatype arrayname [][]

3 What What are preprocessor ? explain any five preprocessor in C

A → A program in C language involves into different preprocessor ...

The C preprocessor is not a part of the compiler but is a separate step in the compilation process. In simple words, a C preprocessor is just a text substitution tool and it instructs the compiler to do required pre-processing

before the actual compilation. All preprocessor commands begin with a hash symbols (#).



list of Preprocessor directives

- # define
- # undef
- # ifdef
- # ifndef
- # if

1) # define

The # define preprocessor directive is use to defined constant or micro substitution. It can use any basic data type

Syntax: # define token value

example:

```
# define # include <stdio.h>
        # define PI . 3.14
         main ( ) {
         printf (" %.6 ", PI);
         }
```

OUTPUT: 3.1400000

2) # undef

The # undef preprocessor directive is used to undefined the constant or macro defined by # define

Syntax: # undef token

example:

```
# include < stdio.h >
# define PI 3.14
# undef PI
main ( ) {
printf ("%.f", PI);
}
```

OUTPUT

Compile time error : 'PI' undeclared

3) # ifdef

The # ifdef preprocessor directive checks if macro is defined by # define. If yes, it executes the code otherwise # else code is executed. if present.

Syntax: # ifdef Macro
// code
endif

Syntax (with # else): # ifdef Macro
// successful code
else
// else code
endif

example :

```
# include < stdio.h >
# include < conio.h >
# define NOINPUT
void main ( ) {
int a=0;
# ifdef NOINPUT
a = 2;
# else
printf (" enter a ");
scanf (" %d",&a);
# endif
printf (" Value of a : %d \n", a);
getch ( );
}
```

OUTPUT

Value of a : 2

4) # & ifndef

The # ifndef preprocessor directive checks if macro is not define by # define . If yes, it executes the code otherwise # else code is executed , if present .

Syntax :

```
# ifndef Macro
// code
# endif
```

Syntax (with # else) :

```
# ifndef Macro
// successful code
# else
// else code
# endif
```

Syntax (with #elif and #else);

```
# if expression
  // if code
# elif expression
// elif code
# else
// else code
# endif
```

Example :

```
# include < stdio.h >
# include < conio.h >
# define NUMBER 0
void main ( ) {
# if(NUMBER == 0)
printf ("Value of Number is : %d", NUMBER);
# endif
getch ( );
}
```

Output :

Value of Number is : 0

4. Write short notes in

a) Nested Structure

b) Array of Structure

c) Unions

d) Self Referential Structure

A a) Nested Structure

Structure can have another structure as a member. There are two ways to define nested structure in C language

1. By Separate Structure

2. By embedded structure

Embedded Structure

We can create two structures but dependent structure should not be used in the main structure

Ex: struct employee

{

int id;

char name[20];

struct data

{

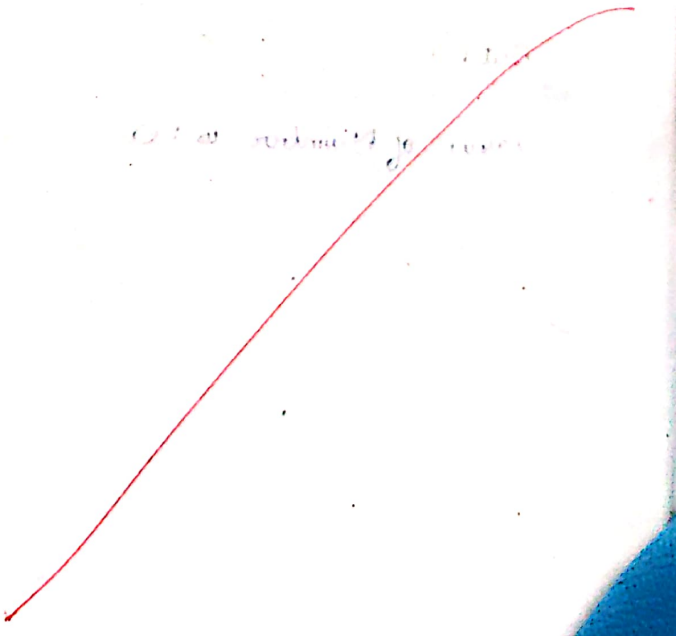
int date;

int month;

int year;

} date of joining;

} employee;



b) Array of Structure

Array of structures to store information of different data types. Each element of the array representing a structure variable. The array of structures is also known as collection of structures

Ex: struct employe emp[5];

Example of structure with array that stores information of 5 students and print it

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

struct student {
int rollno;
char name[10];
};
void main ( ) {
int i;
struct student st[5];
clrscr ( );
printf (" Enter Records of 5 students ");
for (i=0 ; i<5 ; i++) {
printf ("\n Enter Roll no :");
scanf ("%d ", & s[i].rollno );
printf (" \n Enter Name : ");
scanf ("%s", & s[i].name);
}
```

```
printf ("\n Students Information list : ");
for (i=0 ; i<5; i++)
{
printf ("\n Rollno :%d , Name :%s ", s[i].rollno , s[i].name);
}
getch ( );
}
```

c) Unions

- A union is a special data type available in c - language to store different data types in a same memory location
- The system of union is also similar that of structure
- In structure each of member has its own storage location which where as all members of Union use a single memory location which is equal to size of largest data member
  We can access only one member of union at a time.

Syntax

```
Union  Union_name
{
data type    member 1;
data type    member 2;
   :            :
data type    member n;
}
```

ex:

```
Union emp
{
char x;
float y;
}e;
```

d) Self Refurential Structure

- A structure consists of at least a pointer member pointing to the same structure is known as a self refurential structure.
- A self refurential structure is used to create data structures like linked lists, stacks etc.
- Self Refurential structure allows the structure to refer itself by creating a linked data structure

Syntax:

```
struct struct tag-name
    {
    type member 1;
    type member 2;
      :     :
      :     :
    type N    member n;
struct tag-name* name;
}
```

ex:

```
struct emp
{
 int code;
struct emp* name;
}
```

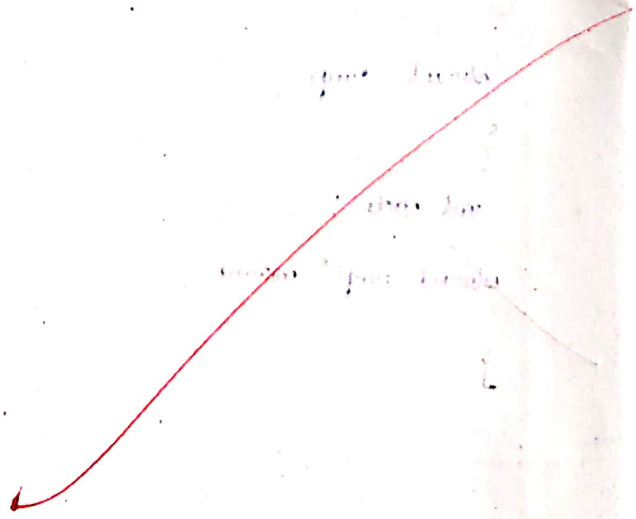
5 What is a structure ? explain the C Syntax of structure declaration with example

A Structure

- It is a user defined data type which is considered constructed by the help of primitive and derived data type
- It can store more than one value of different types of data type of different memory allocation
- The size of structure is the sum of it all data types (or) data type size
- The minimum size of structure is 1 byte
- The structure is declared by the help "struct"
- Every structure should end with semi - column (;)

Syntax:

```
struct structure - name / tag name
{
 data - type   member 1;
 data - type   member 2;
 :
 :
data - type member n;
};
```



Declaration Structure Variable

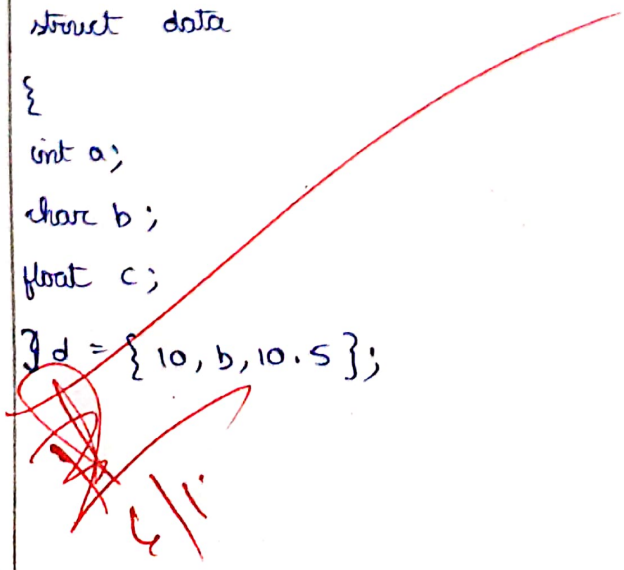
We can declare variable for the structure so that we can access the member of structure easily

Declaration

```
struct structure_name
{
data_type  member 1 ;
data_type  member 2 ;
. . . . .
};
struct struct name
{
data_type  member_name 1 ;
data_type  member_name 2 )
. . . .
} variable 1 , variable 2 , . . . ;
```

Example:

```
struct data
{
int a;
char b ;
float c ;
} d = { 10 , b , 10.5 } ;
```



```
int main( )
{
printf ("value of a is : %d\n", d.a);
printf ("value of b is : %f\n", d.b);
printf ("value of c is : %f\n", d.c);
```

1) Write a C program that defines a structure employee containing the details such as employee Number, employee Name, department name and salary. The structure has