UNIT – 4

PART – A

1.) What is self-referential structure?

A.) Self Referential structures are those structures that have one or more pointers which point to the same type of structure, as their member.
In other words, structures pointing to the same type of structures are self-referential in nature.
Types of Self Referential Structures :-
a.) Self Referential Structure with Single Link
b.) Self Referential Structure with Multiple Links

2.) What is bit-field?

A.) A bit field is a data structure that allows the programmer to allocate memory to structures and unions in bits in order to utilize computer memory in an efficient manner.
Bit fields are of great significance in C programming, because of the following reasons:

- Used to reduce memory consumption.
- Easy to implement.
- Provides flexibility to the code.

3.) Define Structure?

A.) Structure is a collection of one or more variables of different data types that are grouped together under single name.
Ex Program:-

```
struct xyz {
int x;
float y;
char z[10];
};
```

4.) Explain the concept of array of strings using an example?

A.) A string is an array of characters. When we want to store many strings, we need an array of strings. An array of strings in c is a two-dimensional array of characters.
Example Program:-

```
#include <stdio.h>
main()
{
int n;
char names[3][10] = {“Alex”, “Phillip”, “Collins” };
for(n=0; n<3; n++)
printf(“%s \n”,names[n] );
}
```

Output:-

```
Alex
Phillip
Collins
```

5.) Write about enumerated types?

A.) Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.

6.) Write about nested structures?

A.) A nested structure in C is a structure within structure. One structure can be declared inside another structure in the same way structure members are declared inside a structure.

7.) What are command line arguments?

A.) Command line arguments are the arguments which the user gives from the operating system’s command line during the time of execution.

8.) What is pre-processor? Define some pre-processors in c?

A.) Pre-processors are programs that process our source code before compilation. There are 4 main types of preprocessor directives:

- Macros
- File Inclusion
- Conditional Compilation
- Other directives

9.) Differentiate linear and non-linear data structures?

A.)

S.No	Linear Data Structure	Non-Linear Data Structure
1.	In linear data structure, single level is involved.	In non-linear data structure, multiple levels are involved.
2.	Implementation is easy.	Implementation is not easy.
3.	Memory is not utilized in an efficient way.	Memory is utilized in an efficient way.

10.) Briefly discuss about linear data structures?

A.) A linear data structure is a structure in which the elements are stored sequentially, and the elements are connected to the previous and the next element.

Part-B

1.) Write a C program that demonstrates an array of structures?

A.) Program that Demonstrates an array of structures:-

```
#include <stdio.h>
#include <string.h>
struct student{
  int id;
  char name[30];
  float percentage;
};
int main(){
  int i;
  struct student record[2];
        // 1st student's record
  record[0].id=1;
  strcpy(record[0].name, "Bhanu");
  record[0].percentage = 86.5;
      // 2nd student's record
  record[1].id=2;
  strcpy(record[1].name, "Priya");
  record[1].percentage = 90.5;
     // 3rd student's record
  record[2].id=3;
  strcpy(record[2].name, "Hari");
  record[2].percentage = 81.5;
  for(i=0; i<3; i++){
    printf(" Records of STUDENT : %d \n", i+1);
    printf(" Id is: %d \n", record[i].id);
    printf(" Name is: %s \n", record[i].name);
    printf(" Percentage is: %f\n\n",record[i].percentage);
  }
  return 0;
}
```

Output:-

```
Records of STUDENT : 1
Id is: 1
Name is: Bhanu
Percentage is: 86.500000
Records of STUDENT : 2
Id is: 2
Name is: Priya
Percentage is: 90.500000
Records of STUDENT : 3
Id is: 3
Name is: Hari
Percentage is: 81.500000
```

2.) Write a C program that implements students information system?

A.)

```
#include <stdio.h>

struct student

{

char name[50];

int roll;

float marks;

} s[100];

int main()

{

int i,n;

struct student s[100];

printf("Enter total of students:\n");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("\n Enter information of student %d:\n",i+1);

printf("Enter name: ");

scanf("%s", s[i].name);

printf("Enter roll number: ");

scanf("%d", &s[i].roll);

printf("Enter marks: ");

scanf("%f", &s[i].marks);

}
```

```
printf(“Displaying Information:\n”);
for(i=0;i<n;i++)
{
printf(“\n %d no. student info\n”,i+1);
printf(“\tName:%s\n “,s[i].name);
//or use
// puts(s[i].name);
printf(“\t Roll number: %d\n”,s[i].roll);
printf(“\t Marks: %.1f\n\n”,s[i].marks);
}
return 0;
}
```

Output:-

```
Enter total of students:
1
Enter information of student 1:
Enter name: Digvijay
Enter roll number: 1
Enter marks: 66
Displaying Information:
1 no. student info
Name:Digvijay
Roll number: 1
Marks: 66.0
```

3.) Differentiate structre and union with example of each?
A.)

Struct	**Union**
The struct keyword is used to define a structure.	The union keyword is used to define union.
When the variables are declared in a structure, the compiler allocates memory to each variables member. The size of a structure is equal or greater to the sum of the sizes of each data member.	When the variable is declared in the union, the compiler allocates memory to the largest size variable member. The size of a union is equal to the size of its largest data member size.

Changing the value of a member will not affect other variables members.	Changing the value of one member will also affect other variables members.
Each variable member will be assessed at a time.	Only one variable member will be assessed at a time.
The structure allows initializing multiple variable members at once.	Union allows initializing only one variable member at once.
It allows accessing and retrieving any data member at a time.	It allows accessing and retrieving any one data member at a time.
Each variable member occupied a unique memory space.	Variables members share the memory space of the largest size variable.

4.) Explain about declaration, initialization and accessing of a structure with example?

A.) **Declaration:**

We use 'struct' keyword to declare a structure.
Let us declare a student structure containing three fields i.e. name, roll and marks.

```
struct student
{
   char  name[100];
   int   roll;
   float marks;
};
```

Initialization and Accessing:

In C, we initialize or access a structure variable either through dot (.) or arrow (->) operator. This is the most easiest way to initialize or access a structure.

Example:-

```
#include<stdio.h>
struct Vehicle
{
int wheels;
char vname[20];
char color[10];}v1 = {4," Swift Desire "," White "};
int main(){
printf(Vehicle No of Wheels : %d",v1.wheels);
printf(Vehicle Name : %s",v1.vname);
printf(Vehicle Color : %s",v1.color);
return(0);
}
```

Output:

```
Vehicle No of Wheels : 4
Vehicle Name : Swift Desire
Vehicle Color : White
```

5.) What is structure? Define a structure called salary and another structure called allowance. Use the structure variable allowance in salary structure and write a program to read data into the structure variables.?

A.) **Structure :** Structure is a collection of one or more variables of different data types that are grouped together under single name.

Program:

```
#include <stdio.h>
struct allowance
{
```

```
  int a;
 };
  struct salary
 {
  int s;
 struct allowance a1;
 };
 void main()
 {
 struct salary s1;
 printf("Eter salary= ");
 scanf("%d",&s1.s);
 printf("Enter allowance = ");
 scanf("%d",&s1.a1.a);
 printf("salary=%d allowance= %d", s1.s, s1.a1.a);
 }
```

Output:

```
Enter salary=30000
Enter allowance=2000
Salary=30000  Allowance=2000
```

6.) Explain the concept of array of structures?

A.) An array of structure in C programming is a collection of different datatype variables, grouped together under a single name.

General form of structure declaration:

```
struct tagname{
 datatype member1;
  datatype member2;
 datatype member n;
};
```

Example:

The following **program** shows the usage of array of structures.

```
#include <stdio.h>
#include <string.h>
struct student{
   int id;
   char name[30];
   float percentage;
};
int main(){
   int i;
   struct student record[2];
   // 1st student's record
   record[0].id=1;
   strcpy(record[0].name, "Bhanu");
```

```
  record[0].percentage = 86.5;
  // 2nd student's record
  record[1].id=2;
  strcpy(record[1].name, "Priya");
  record[1].percentage = 90.5;
  // 3rd student's record
  record[2].id=3;
  strcpy(record[2].name, "Hari");
  record[2].percentage = 81.5;
  for(i=0; i<3; i++){
    printf(" Records of STUDENT : %d \n", i+1);
    printf(" Id is: %d \n", record[i].id);
    printf(" Name is: %s \n", record[i].name);
    printf(" Percentage is: %f\n\n",record[i].percentage);
  }
  return 0;
}
```

Output:

```
Records of STUDENT : 1
Id is: 1
Name is: Bhanu
Percentage is: 86.500000
Records of STUDENT : 2
Id is: 2
Name is: Priya
Percentage is: 90.500000
Records of STUDENT : 3
Id is: 3
Name is: Hari
Percentage is: 81.500000
```

7.) Write a short note on typedef and enum with examples?

A.) **Typedef:**

The typedef is a keyword used in C programming to provide some meaningful names to the already existing variable in the C program. It behaves similarly as we define the alias for the commands. In short, we can say that this keyword is used to redefine the name of an already existing variable.

Example Program:

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

```
{
typedef unsigned int unit;
unit i,j;
i=10;
j=20;
printf("Value of i is :%d",i);
printf("\nValue of j is :%d",j);
return 0;
}
```

Output:

Value of i is :10
Value of j is :20

Enum :

Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.

Example Program :

```
#include<stdio.h>
enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};
int main()
{
   enum week day;
   day = Wed;
   printf("%d",day);
   return 0;
}
```

Output:

2

8.) Write a program to handle students profile using data structures.?

A.) **Program to handle students profile using data structures:**

```
#include <stdio.h>
struct student {
   char firstName[50];
   int roll;
   float marks;
} s[5];

int main() {
   int i;
   printf("Enter information of students:\n");

   // storing information
   for (i = 0; i < 5; ++i) {
      s[i].roll = i + 1;
      printf("\nFor roll number%d,\n", s[i].roll);
      printf("Enter first name: ");
      scanf("%s", s[i].firstName);
      printf("Enter marks: ");
      scanf("%f", &s[i].marks);
   }
   printf("Displaying Information:\n\n");

   // displaying information
   for (i = 0; i < 5; ++i) {
      printf("\nRoll number: %d\n", i + 1);
      printf("First name: ");
      puts(s[i].firstName);
```

```
        printf("Marks: %.1f", s[i].marks);
        printf("\n");
    }
    return 0;
}
```

Output:

```
Enter information of students:
For roll number1,
Enter name: Tom
Enter marks: 98
For roll number2,
Enter name: Jerry
Enter marks: 89
.
.
.
Displaying Information:
Roll number: 1
Name: Tom
Marks: 98
```