MODULE 4 SOLUTIONS

PART A

1) Define the term pointer and state the applications?

ANS) Pointer is a variable which stores the address of another variable. Since Pointer is also a kind of variable, this pointer itself will be stored at different memory locations.

APPLICATIONS OF POINTERS:

- To pass arguments by reference
- For accessing array elements
- To return multiple values
- Dynamic memory allocation

2) Define generic pointers and Null pointers in C.

ANS) Null pointer: A pointer variable which is initialized with null values is known as Null pointer.

Generic pointers: A pointer that has no associated data type with it. A void pointer can hold addresses of any type and can be type casted to any type. Generic pointers are also called Void pointers.

3) List the functions used for dynamic memory allocation in C.

ANS) Functions used for dynamic memory allocation in C are:

i) malloc()

ii) calloc()

iii) free()

iv) realloc()

To use these functions we have to initialize #incude <stdlib.h> header file.

i) malloc(): This function is used to dynamically allocate a single large block of memory with specified size

ii) calloc(): This function is used to dynamically allocate the specified number of blocks of memory of specified type. It initialize each block with a default value

iii) free(): The memory allocated using function malloc() and calloc() is not deallocated on its own. Hence the free() method is used wherever the dynamic allocation takes place. It helps to reduce wastage of memory by freeing it.

iv) realloc(): The memory allocated previously in dynamic memory is now reallocated and stored in another location is known as realloc().

4) Explain pointer to pointer in C.

ANS) Pointer to pointer means a chain of pointers. Normally, a pointer contains the address of a variable. When we define a pointer to a pointer, the first pointer contains the address of the second pointer, which points to the location that contains the actual value.

5) Explain bit fields in C.

ANS) The variables defined with a predefined length are called bit fields. A bit field can hold more than a single bit. By using bit fields it makes the code flexible, the program becomes more organized and structured, Hence it reduces the wastage of memory.

6) Explain Preprocessor directives with examples

ANS) A Preprocessor tells the compiler to preprocess the source code before compiling. A C-program starts with the preprocessor directive i.e., #include, #define, #undef, etc. By using #include we can link the header files that are needed to use the functions. By using #define we can define some constants.

7) Define the term structure and state how the members of a structure are accessed.

ANS) As we know, an array is a collection of elements of the same datatype, but many times we have to store the elements of different data types. To do so STRUCTURES is a collection of elements of all different data types in the same variable. Each member declared in structures is called as members, struct is a key word which is used to declare structures. Members of structures are enclosed within open and closed braces.

Member of structures are accessed by :

- Array elements are accessed using the subscripts variables, similarly Structure members are accessed by using dot (.) operator
- (.) is called a "Structure member operator".
- Use this Operator in between "Structure name" & "Member name".

8) Compare the differences between arrays and structures

ANS)

ARRAYS	STRUCTURES
•An array is defined as the collection of similar type of data items stored at contiguous memory locations •Array uses subscripts or “[ ]” (square bracket) for element access •Array is a pointer as it points to the first element of the collection. •Instantiation of Array objects is not possible. •Array size is fixed and is basically the number of elements multiplied by the size of an element. •Bit field is not possible in an Array. •Array declaration is done simply using [] and not any keyword. •Arrays is a primitive data type •Array traversal and searching is easy and fast. •data_type array_name[size]; •Array elements are stored in contiguous memory locations. •Array elements are accessed by their index number using subscripts.	•Structure refers to a collection consisting of elements of heterogeneous data type. •Structure uses “.” (Dot operator) for element access •Structure is not a pointer •Instantiation of Structure objects is possible. •Structure size is not fixed as each element of Structure can be of different type and size. •Bit field is possible in a Structure. •Structure declaration is done with the help of the “struct” keyword. •Structure is a user-defined datatype. •Structure traversal and searching is complex and slow. •struct sruct_name{ data_type1 ele1; data_type2 ele2; }; •Structure elements may or may not be stored in a contiguous memory location. •Structure elements are accessed by their names using dot operators.

9) Explain nested structure in C.

ANS) Nested structures in C are nothing but structure within structure. One structure can be declared inside another structure as we declare structure member inside a structure

EX: **SOURCE CODE:**

```
#include<stdio.h>
struct address   {
    char city[20];
    int pin;
    char phone[14];
};
struct employee  {
    char name[20];
    struct address add;
};
void main ()  {
    struct employee emp;
    printf("Enter employee information?\n");
    scanf("%s %s %d %s",emp.name,emp.add.city, &emp.add.pin, emp.add.phone);
    printf("Printing the employee information....\n");
    printf("name: %s\nCity: %s\nPincode: %d\nPhone:
%s",emp.name,emp.add.city,emp.add.pin,emp.add.phone);
}
```

OUTPUT:

```
Enter employee information?

SATISH   HYDERABAD   500072    950****898

Printing the employee information....

name: SATISH

City: HYDERABAD

Pincode: 500072

Phone: 950****898
```

10) Compare the differences between structure and union.

ANS)

	STRUCTURE	UNION
Keyword	The keyword struct is used to define a structure	The keyword union is used to define a union
size	When a variable is associated with a structure. The compiler allocates the	When a variable is associated with a union, the compiler allocates the

	memory for each member. The size of structures is greater than or equal to the sum of sizes of its members	memory by considering the size of the largest memory. So size of union is equal to the size of largest member
memory	Each memory within a structure is assigned a unique storage area of location.	Memory allocated is shared by individual members of union
Value altering	Altering the value of a member will not affect other member of the structure	Altering the value of any of the member will affect other member values
Accessing members	Individual member can be accessed at a time	Only one member can be accessed at a time
Initialization of members	Several members of a structure can initialize at once	Only the first members of a union can be initialized

11) Explain the array of structures in C.

ANS) An array of structures in C programming is a collection of different data type variable, grouped together under a single name

- The most common use of structures in c programming is an array of structures
- To declare an array of structures, first the structure must be defined and then an array variable of that type should be defined
- for example – struct book b[10]; //10 elements in an array of structures of type “book”

EX:

SOURCE CODE:

```
#include <stdio.h>
#include <string.h>
struct student{
    int rollno;
    char name[10];
};
int main(){
    int i;
    struct student st[5];
    printf("Enter Records of 5 students");
    for(i=0;i<5;i++){
        printf("\nEnter Rollno:");
        scanf("%d",&st[i].rollno);
        printf("\nEnter Name:");
        scanf("%s",&st[i].name);
    }
```

```
    printf("\nStudent Information List:");
    for(i=0;i<5;i++){
    printf("\nRollno:%d, Name:%s",st[i].rollno,st[i].name);
    }
    return 0;
}
```

OUTPUT:

```
Enter Records of 5 students
Enter Rollno:1
Enter Name:Sonoo
Enter Rollno:2
Enter Name:Ratan
Enter Rollno:3
Enter Name:Vimal
Enter Rollno:4
Enter Name:James
Enter Rollno:5
Enter Name:Sarfraz

Student Information List:
Rollno:1, Name:Sonoo
Rollno:2, Name:Ratan
Rollno:3, Name:Vimal
Rollno:4, Name:James
```

12) Write about enumerated data types.

ANS)Enumerate (enum) is a user defined datatype in C. It is mainly used to assign names to integral constants, which makes a program easy to read and maintain. The Keyword "enum" is used to declare an enumeration

EX:

SOURCE CODE:

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

```
}
```

OUTPUT:

```
DAY=2
```

13) State the default starting values of enumerated set

ANS) If we did not assign a starting value to enum names, the compiler would assign 0 as the starting value of the enumerated set by default.

14) Explain the usage of typedef with example

ANS) In C programming language provides a key word called typedef, which provides some meaningful names to the already existing variable in C program. In short we can say that this keyword is used to redefine the name of an already existing variable.

Syntax: typedef<existing name> <alias name>

EX:**SOURCE CODE:**

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

15) State how to access the members of the structure in C.

ANS) Member of structures are accessed by :

- Array elements are accessed using the subscripts variables, similarly Structure members are accessed by using dot (.) operator
- (.) is called a "Structure member operator".
- Use this Operator in between "Structure name" & "Member name".

16) Explain Pointers as functions arguments with example

ANS) Pointer as a Function argument(parameter) is used to hold the address of argument passed during function call. This is also known as call by reference. When a function is called by reference if any changes made to the reference variable this will effect the original variable

EX:

SOURCE CODE:

```
#include <stdio.h>
void swap(int *a, int *b);
int main() {
    int m = 10, n = 20;
    printf("m = %d\n", m);
    printf("n = %d\n", n);
    swap(&m, &n);    //passing address of m and n to the swap function
    printf("After Swapping:\n");
    printf("m = %d\n", m);
    printf("n = %d", n);
    return 0;
}
void swap(int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

OUTPUT:

```
m = 10
n = 20
After Swapping:
m = 20
n = 10
```

17) Predict the output of the following code.

SOURCE CODE:

```
struct {
    int i;
    float f;
}
var;
void main() {
    var.i=5;
    var.f=9.76723;
    printf("%d %.2f",var.i,var.f);
}
```

OUTPUT:

```
5 9.77 (it will raise a warning include<stdio.h> or provide a  declaration of printf)
```

18) Predict the output of the following code.

SOURCE CODE:

```
#include<stdio.h>
struct values
{
    int i; float f;
};
void main()
{
    struct values var={555,67.05501};
    printf("%2d%.2f",var.i,var.f);
}
```

OUTPUT:

```
55567.06
```

19) Consider the following C declaration and find the size required by the structure. Assume that objects of the type short float and long occupy 2 bytes, 4 bytes and 8 bytes, respectively. The Memory for variable t will be:

SOURCE CODE:

```
struct {
short s[5];
union {
    float y; long z;
    } u;
} t;
```

ANS) Short array s[5] will take 10 bytes as size of short is 2 bytes. When we declare a union, memory allocated for the union is equal to memory needed for the largest member of it, and all members share this same memory space. Since u is a union, memory allocated to u will be max of float y(4 bytes) and long z(8 bytes). So, the total size will be 18 bytes (10 + 8).

20) Predict the output of following C program

SOURCE CODE:

```
#include<stdio.h>
struct Point {
    int x, y,z; //To pass arguments by reference

int main() {
    struct Point p1 = {.y = 0, .z = 1, .x =2};
    printf("%d %d %d",p1.x, p1.y, p1.z);
    return 0;
}
```

OUTPUT:

`Error` (To pass arguments by reference)

PART B

1) Write a C program to read your full name, Date of birth and display the same using the concept of nested structure.

SOURCE CODE:

```
#include<stdio.h>
struct student  {
    char name[10];
    struct dob    {
        int dd;
        int mm;
        int yr;
    }DOB;
};
int main() {
    struct student s;
    printf("enter name: ");
    scanf("%s",&s.name);
    printf("enter DOB[DD MM YYYY]:");
    scanf("%d%d%d",&s.DOB.dd,&s.DOB.mm,&s.DOB.yr);
    printf("student name: %s\n",s.name);
    printf("DOB:%d/%d/%d",s.DOB.dd,s.DOB.mm,s.DOB.yr);
}
```

OUTPUT:

```
enter name: Satish
enter DOB[DD MM YYYY]:20 12 2001
student name: Satish
DOB:20/12/2001
```

2) Write a C program to maintain a book structure containing name, author and pages as structure members. Pass the address of structure variable to a user defined function and display the contents

SOURCE CODE:

```
#include<stdio.h>
#define SIZE 20
struct structure{
```

```
    char name[SIZE];
    char author[SIZE];
    int pages;
};
void output(struct structure a[],int n);
void main()     {
    struct structure b[SIZE];
    int n,i;
    printf("Enter the Numbers of Books:");
    scanf("%d",&n);
    printf("\n");
    for(i=0;i<n;i++)
    {
        printf("\tBook %d Detail\n",i+1);
        printf("Enter the Book Name:");
        scanf("%s",b[i].name);
        printf("Enter the Author of Book:");
        scanf("%s",b[i].author);
        printf("Enter the Pages of Book:");
        scanf("%d",&b[i].pages);
    }
    output(b,n);
}
void output(struct structure a[],int n)     {
    int i,t=1;
    for(i=0;i<n;i++)    {
        printf("\n");
        printf("Book No.%d\n",t);
        printf("%dst Book Name is: %s \n",t,a[i].name);
        printf("%dst Book Author is: %s \n",t,a[i].author);
        printf("%dst Book Pages is: %d",t,a[i].pages);
        printf("\n");
        t++;
    }
}
```

OUTPUT:

```
Enter the Numbers of Books:2

Book 1 Detail

Enter the Book Name:C Language

Enter the Author of Book:Yashwant Kanetkar

Enter the Pages of Book:264
```

```
        Book 2 Detail
Enter the Book Name:LET_US_PYTHON
Enter the Author of Book:Yashwant Kanetkar
Enter the Pages of Book:500


Book No.1
1st Book Name is: C Language
1st Book Author is: YashavantKanetkar
1st Book Pages is: 264


Book No.2
2st Book Name is: LET_US_PYTHON
2st Book Author is: YashavantKanetkar
2st Book Pages is: 500
```

3) A marketing company has 50 employees and it maintains employee records in terms of their empid, empname, desg, salary, quantity, sales amount. The company gives a 10% hike in salary to the employees if their sales amount is more than 50000/-. Write a C program that displays the employee records who got a hike in salary.

SOURCE CODE:

```
#include<stdio.h>
struct market   {
      int empid,salary,quantity,salesamount;
      char empname[20],desg[20];
};
main()     {

      int i,n;
      printf("enter the no.of check");
      scanf("%d",&n);
      struct market emp[50];
      for(i=0;i<50;i++)     {
            printf("enter %dst emp details:\n",i+1);
            printf("Enter id: ");
            scanf("%d",&emp[i].empid);
            printf("Enter name: ");
            scanf("%s",&emp[i].empname);
            printf("enter designation: ");
            scanf("%s",&emp[i].desg);
            printf("enter salary: ");
```

```
            scanf("%d",&emp[i].salary);
            printf("enter quantity: ");
            scanf("%d",&emp[i].quantity);
            printf("Enter sales amount: ");
            scanf("%d",&emp[i].salesamount);
      }
      printf("Emp who will get hike:");
      for(i=0;i<50;i++)     {
            if(emp[i].salesamount>50000)          {
                  printf("%d\n",emp[i].empid);
                  printf("%s\n",emp[i].empname);
                  printf("%s\n",emp[i].desg);
                  printf("%d\n",emp[i].salary);
                  printf("%d\n",emp[i].quantity);
                  printf("%d\n",emp[i].salesamount);
            }
      }
}
```

OUTPUT:

```
enter the no.of check2
enter 1st emp details:
Enter id: 92
Enter name: amar
enter designation: salesmanager
enter salary: 80000
enter quantity: 100
Enter sales amount: 60000
enter 2st emp details:
Enter id: 103
Enter name: abhay
enter designation: salesmen
enter salary: 50000
enter quantity: 80
Enter sales amount: 50000
Emp who will get hike:
92
amar
salesmanager
```

```
80000

100

60000
```

4) IARE College is maintaining student attendance records by storing rollno, stdname, attendance percentage in 5 different subjects. Write a C program using structures to find the average attendance percentage and print the following

a. If attendance percentage >=75 then the print student is eligible for writing the final exam.

b. If attendance percentage >= 65 and <75 then print student is in the condonation list.

c. Otherwise not eligible for writing exams

SOURCE CODE:

```
enter no of students:2

Enter rno:92
```

```c
#include<stdio.h>
struct IARE {
int rno;
char std[20];
int atten[5];
};
int main() {
      int n,i,j,sum,avg;
      printf("enter no of students:");
      scanf("%d",&n);
      struct IARE s[n];
      for(i=0;i<n;i++) {
            printf("Enter rno:");
            scanf("%d",&s[i].rno);
            printf("Enter name:");
            scanf("%s",&s[i].std);
            for(j=0;j<5;j++)      {
                  printf("Enter sub %d:",j+1);
                  scanf("%d",&s[i].atten[j]);
            }
      }
      for(i=0;i<n;i++) {
            sum=0;
            for(j=0;j<5;j++) {
                  sum=sum+s[i].atten[j]; }
            avg=sum/5;
            if(avg>=75)
                printf("Eligible\n");
```

```
            else if(avg>=65 && avg<75)
                printf("conditionality\n");
            else
                printf("Not\n");
        }
 }
```

OUTPUT:

```
Enter name:amar
Enter sub 1:92
Enter sub 2:95
Enter sub 3:80
Enter sub 4:79
Enter sub 5:90
Enter rno:90
Enter name:abhay
Enter sub 1:90
Enter sub 2:80
Enter sub 3:85
Enter sub 4:55
Enter sub 5:63
Eligible
Not
```

5) Consider the declaration of the structure

```
typedef struct
{
char x; char *y;
int z[20];
} status;
```

Discuss whether the following are valid, if invalid, give reason.

a) struct statuss1; b) struct statuss2[25]; c) statuss3; d) status s4[20];

6) Compare and explain the following with suitable examples:

a) Nested Structures

b) Array of structures

ANS) a) Nested Structures:

- Nested structure in C is nothing but structure within structure. One structure can be declared inside another structure as we declare structure members inside a structure
- The structure variable can be a normal structure variable or a pointer variable to access the data.
- Structure within structures in C can use normal variable or pointer variable

EX:

SOURCE CODE:

```
#include<stdio.h>
struct address   {
    char city[20];
    int pin;
    char phone[14];
    };
    struct employee  {
        char name[20];
        struct address add;
}emp;
void main ()  {
    printf("Enter employee information?\n");
    scanf("%s %s %d %s",emp.name,emp.add.city, &emp.add.pin, emp.add.phone);
    printf("Printing the employee information....\n");
    printf("name: %s\nCity: %s\nPincode: %d\nPhone:
%s",emp.name,emp.add.city,emp.add.pin,emp.add.phone);
}
```

```
OUTPUT:

Enter employee information?

IARE

HYDERABAD

500043

9876543215

Printing the employee information....
```

```
name: IARE

City: HYDERABAD

Pincode: 500043

Phone: 9876543215
```

b) Array of structures: An array of structures in C can be defined as the collection of multiple structure variables where each variable contains information about different entities. The array of structures in C are used to store information about multiple entities of different data types. The array of structures is also known as the collection of structures.

SAMPLE SYNTAX EXAMPLE:

```
struct employee
{
    int id;
    char name[5];
    float salary;
};
struct employee emp[2];
```

sizeof (emp) = 4 + 5 + 4 = 13 bytes

sizeof (emp[2]) = 26 bytes

EX:

SOURCE CODE:

```
#include<stdio.h>
struct student {
    int rollno;
    char name[10];
};
int main() {
    int i;
    struct student st[3];
    printf("Enter Records of 3 students\n");
    for(i=0;i<3;i++)    {
        printf("Enter Rollno: ");
        scanf("%d",&st[i].rollno);
        printf("Enter Name: ");
        scanf("%s",&st[i].name);
    }
    printf("Student Information List:");
    for(i=0;i<3;i++)
        printf("\nRollno: %d, Name: %s",st[i].rollno,st[i].name);
   return 0;
}
```

OUTPUT:

```
Enter Records of 3 students
```

```
Enter Rollno: 1
Enter Name: IARE
Enter Rollno: 2
Enter Name: IT
Enter Rollno: 3
Enter Name: CSE
Student Information List:
Rollno: 1, Name: IARE
Rollno: 2, Name: IT
Rollno: 3, Name: CSE
```

7) Explain the following with suitable example:

a) self-referential structures

b) enumerated types

ANS) i) Self-referential structures: these are the structures that have one or more pointers which point to the same type of structure, as their member. In other word, structure pointing to the same type of structures are self-referential in nature

OR

A structure that contains pointers to structures of its own type is known as self-referential structures.

Declaration of self-referential Structure:

```
Struct <structure_name>
{
      Structure element 1;
      ..............................
      Structure element n;
      Struct <structure_name>*pointer_name;
};
Struct <structure name> variable1,variable 2, ...;
```

b) enumerated types : Enumeration is a user defined datatype in C lang. it is used to assign names to the integral constants which makes a program easy to read and maintain. The keyword ”enum” is used to declare an enumeration

```
SYNTAX: enum enum_name{const1, const2, ...};
```

EX:

SOURCE CODE:

```
#include<stdio.h>
enum week{Mon=10, Tue, Wed, Thur, Fri=10, Sat=16, Sun};
enum day{Mond, Tues, Wedn, Thurs, Frid=18, Satu=11, Sund};
int main() {
   printf("The value of enum week: %d %d %d %d %d %d %d\n",Mon , Tue, Wed, Thur,
Fri, Sat, Sun);
   printf("The default value of enum day: %d %d %d %d %d %d %d",Mond , Tues,
Wedn, Thurs, Frid, Satu, Sund);
   return 0;
}
```

OUTPUT:

```
The value of enum week: 10 11 12 13 10 16 17

The default value of enum day: 0 1 2 3 18 11 12
```

8) Write a C program to pass a copy of the entire structure named stores containing members' product-name, price and quantity to a function.

SOURCE CODE:

```
#include <stdio.h>
#include <string.h>
 struct stores
{
    char product_name[20];
    float price,quantity;
};

void func(struct stores record);
int main()
{
    struct stores record;
    record.quantity=2;
    strcpy(record.product_name,"laptop");
    record.price=65000;
    func(record);
    return 0;
}
void func(struct stores record)
{
```

```
    printf("Name is: %s \n", record.product_name);
    printf("quantity is: %.2f \n", record.quantity);
    printf("Price is: %.2f \n", record.price);
}
```

OUTPUT:

```
Name is: laptop
quantity is: 2.00
Price is: 65000.00
```

11) Write a C program to maintain a record of n student details using an array of structures with four fields (rollno, name, marks and grade). Assume appropriate data type for each field. Print the marks of the student name as input.

Understand

SOURCE CODE:

```
#include<stdio.h>
struct student
{
    char name[20],grade[2];
     int rollno;
    float marks;
};
int main( )
{
    int i,n;
    printf("Enter how many records u want to store :: ");
    scanf("%d",&n);
    struct student stuarr[n];
    printf("Enter name, roll no. and marks Below :: \n");
    for(i=0; i<n; i++)
    {
     printf("\nEnter %d record :: \n",i+1);
        printf("Enter Name :: ");
        scanf("%s",stuarr[i].name);
        printf("Enter RollNo. :: ");
        scanf("%d",&stuarr[i].rollno);
        printf("Enter Marks out of 100:: ");
        scanf("%f",&stuarr[i].marks);
        printf("enter Grade::");
```

```
        scanf("%s",stuarr[i].grade);
    }
    printf("\n\tName\tRollNo\tMarks\tGrade\n");
    for(i=0; i<n; i++)
        printf("\t%s\t%d\t%.2f\t%s\n", stuarr[i].name, stuarr[i].rollno,
 stuarr[i].marks,stuarr[i].grade);
    return 0;
 }
```

OUTPUT:

```
Enter how many records u want to store :: 2
Enter name, roll no. and marks Below ::

Enter 1 record ::
Enter Name :: SATISH
Enter RollNo. :: 92
Enter Marks out of 100:: 90
enter Grade::A

Enter 2 record ::
Enter Name :: sbhay
Enter RollNo. :: 95
Enter Marks out of 100:: 89
enter Grade::B

        Name    RollNo  Marks   Grade
        SATISH  92      90.00   A
        sbhay   95      89.00   B
```

12) Define a structure called complex consisting of two floating point numbers x and y and declare a variable p of type complex. Assign initial values 0.0 and 1.1 to the members.

SOURCE CODE:

```
#include<stdio.h>
struct complex
```

```
{
float x;
float y;
};
int main()
{
struct complex p = {0.0,1.1};
printf("%f\n%f",p.x,p.y);
}
```

OUTPUT:

```
0.000000
1.100000
```

13) Define a structure data type called time struct containing 3 members integer hour, integer minute and integer second. Develop a program that would assign values to the individual members and display the time in the following format:16 : 40 : 51

SOURCE CODE:

```
#include <stdio.h>
struct time_struct
{
      int hour,minute,second;
}t;
int main(void)
{
      printf("Enter Hour : ");
      scanf("%d",&t.hour);
      printf("Enter Minute: ");
      scanf("%d",&t.minute);
      printf("Enter Second : ");
      scanf("%d",&t.second);
      printf("Time %d:%d:%d",t.hour%24,t.minute%60,t.second%60);
      return 0;
}
```

OUTPUT:

```
Enter Hour : 24
Enter Minute: 12
Enter Second : 12
Time 0:12:12
```

14) Define a structure named census with the following 3 members:

a. A character array city[ ] to store names.

b. A long integer to store the population of the city.

c. A float member to store the literacy level. Write a program to do the following:

d. To read details for 5 cities randomly using an array variable.

e. To sort the list alphabetically.

f. To sort the list based on literacy level.

g. To sort the list based on population.

h. To display sorted lists.

SOURCE CODE:

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

struct census {
  char city[20];
  long int pop;
  float literacy;
}city_names[5];

//sorting alphabetically!
void sort_alpha(struct census x[]){
  for(int i=0;i<5;i++){
    for(int j=i+1;j<5;j++)
      if(strcmp(x[i].city,x[j].city)>0){
        struct census temp = x[i];
        x[i] = x[j];
        x[j] = temp;
    }
  }

  printf("Display after sorting Alphabetically:\n");
  for(int i=0;i<5;i++){
    printf("%s %li %f\n",x[i].city,x[i].pop,x[i].literacy);
  }
}
```

```c
//Sorting based on literacy:
void sort_lit(struct census x[]){
  for(int i=0;i<5;i++){
    for(int j=i+1;j<5;j++){
      if(x[i].literacy>x[j].literacy){
        struct census temp = x[i];
        x[i] = x[j];
        x[j] = temp;
    }
  }
  }
  printf("\nDisplay after sorting based on literacy:\n");
  for(int i=0;i<5;i++){
    printf("%s %li %f\n",x[i].city,x[i].pop,x[i].literacy);
  }
}

//Sorting based on Population:
void sort_pop(struct census x[]){
  for(int i=0;i<5;i++){
    for(int j=i+1;j<5;j++){
      if(x[i].pop>x[j].pop){
        struct census temp = x[i];
        x[i] = x[j];
        x[j] = temp;
    }
  }
  }
  printf("\nDisplay after sorting based on population:\n");
  for(int i=0;i<5;i++){
    printf("%s %li %f\n",x[i].city,x[i].pop,x[i].literacy);
  }
}


int main(){
  for(int i=0;i<5;i++){
    printf("\nEnter City name, population no. and literacy level: ");
    scanf("%s %li
%f",city_names[i].city,&city_names[i].pop,&city_names[i].literacy);
  }
```

```
  sort_alpha(city_names);
  sort_lit(city_names);
  sort_pop(city_names);

  return 0;
}
```

OUTPUT:

```
enter city name, population and literacy level:HYDERABAD 180000 75
enter city name, population and literacy level:CHENNAI 300000 75
enter city name, population and literacy level:ONGOLE 80000 60
enter city name, population and literacy level:MUMBAI 500000 80
enter city name, population and literacy level:DELHI 300000 80
sorted order are
ONGOLE  80000   60.000000
HYDERABAD       180000  75.000000
CHENNAI 300000  75.000000
MUMBAI  500000  80.000000
DELHI   300000  80.000000
```

15) Define a structure that can describe a hotel. It should have members that include the name, address, grade, average room charge, and number of rooms. Write functions to perform the following operations:

a. To print out hotels of a given grade in order of charges.

b. To print out hotels with room charges less than a given value.

SOURCE CODE:

```
#include<stdio.h>
struct hotel
{
    char name[20],add[20];
    int grade,arc,rooms;
};
void output();
```

```
void out();
struct hotel inn[]={
    {"PLAZA","G-99,DELHI",3,4500,50},
    {"MYUR","E-45,NOIDA",4,5000,100},
    {"RAJDOOT","H-44,DELHI",2,4000,50},
    {"SAMRATH","B-75,BOMBAY",5,6000,200},
    {"SURYA","A-77,NOIDA",1,3500,150}
};
void main()
{
    int go;
    printf("Enter 1 for grade search\n");
    printf("Enter 2 to search by charge:");
    scanf("%d",&go);
    switch(go)
    {
        case 1:
            output();
            break;
        case 2:
            out();
            break;
        default:
            printf("Wrong input");
            break;
    }
}
void output()
{
    int gr,i;
    printf("Enter Grade 1 to 5:");
    scanf("%d",&gr);
    if(gr>=1||gr<=5)
    {
        for(i=0;i<=4;i++)
        {
            if(inn[i].grade==gr)
                printf("Hotel Name: %s\nAddress:%s\nGrade:%d\nAverage Room
charge:%d\nNumber of
Rooms:%d",inn[i].name,inn[i].add,inn[i].grade,inn[i].arc,inn[i].rooms);
        }
    }
    else
```

```
        printf("Wrong grade input!");
}
void out()
{
    int ch,i=0;
    printf("Enter the Room charges not greater than 6000:");
    scanf("%d",&ch);
    while(i<5)
    {
        if(inn[i].arc<ch)
            printf("Hotel Name: %s\nAddress:%s\nGrade:%d\nAverage Room
charge:%d\nNumber of
Rooms:%d\n",inn[i].name,inn[i].add,inn[i].grade,inn[i].arc,inn[i].rooms);
        i++;
    }
}
```

OUTPUT:

```
Enter 1 for grade search

Enter 2 to search by charge:1

Enter Grade 1 to 5:3

Hotel Name: PLAZA

Address:G-99,DELHI

Grade:3

Average Room charge:4500

Number of Rooms:50
```

16)Define a structure called cricket that will describe the following information: Player name ,Team name ,Batting average using cricket, declare an array play program to read the information about all the 50 players and print a teamwise with their batting average.

SOURCE CODE:

```
#include<stdio.h>
struct cricket
{
      char pname[20];
      char tname[20];
      float bavg;
};
```

```c
int main()
{
	struct cricket s[3],t;
	int i,j,n;
	printf("enter no of players list to be stored:");
	scanf("%d",&n);
	float p;
	printf("Enter data of %d players\n",n);
	for(i=0;i<n;i++)
	{
		printf("Enter PName TName BAvg for player %d = ",i+1);
		scanf("%s %s %f",s[i].pname,s[i].tname,&p);
		s[i].bavg=p;
	}
	for(i=1;i<=n-1;i++)
	{
		for(j=1;j<=n-i;j++)
		{
			if(strcmp(s[j-1].tname,s[j].tname)>0)
			{
				t=s[j-1];
				s[j-1]=s[j];
				s[j]=t;
			}
		}
	}
	printf("\nAfter teamwise sorting... Player list is ");
	for(i=0;i<n;i++)
		printf("\n%s\t %s\t %.2f",s[i].pname,s[i].tname,s[i].bavg);
	return 0;
}
```

OUTPUT:

```
enter no of players list to be stored:3
Enter data of 3 players
Enter PName TName BAvg for player 1 = DHONI INDIA 95
Enter PName TName BAvg for player 2 = KOHLI INDIA 90
Enter PName TName BAvg for player 3 = ROHITSHARMA INDIA 89

After teamwise sorting... Player list is
DHONI    INDIA   95.00
```

```
KOHLI     INDIA    90.00
ROHITSHARMA       INDIA    89.00
```

17) IARE maintains the salary details of every employee by storing their name, department, basic pay, da, hra and cca. Store this information in a nested structure and display the salary of an employee.

SOURCE CODE:

```
#include<stdio.h>
struct s1
{
      char *name;
      char *dep;
};
struct s2
{
      float bp,da,hra,cca,total;
      struct s1 g;
};
int main()
{
      int n,i,j;
      printf("enetr no of employess:");
      scanf("%d",&n);
      struct s2 f[n];
      for(i=0;i<n;i++)
      {
          printf("enetr name,department,bs,DA,hra,cca of %dst: ",i+1);
            scanf("%s",&f[i].g.name);
            scanf("%s",&f[i].g.dep);
            scanf("%f",&f[i].bp);
            scanf("%f",&f[i].da);
            scanf("%f",&f[i].hra);
            scanf("%f",&f[i].cca);
      }
      for(i=0;i<n;i++)
      {
            f[i].total=0;

f[i].total=f[i].bp+(f[i].da/100*f[i].bp)+(f[i].hra/100*f[i].bp)+(f[i].cca/100*f[
i].bp);
```

```
        printf("gross salary=%.2f\n",f[i].total);
      }
 }
```

OUTPUT:

```
enetr no of employess:2
enetr name,department,bs,DA,hra,cca of 1st: AMAR ECE 180000 10 12 16
enetr name,department,bs,DA,hra,cca of 2st: PADMAJA CSE 150000 10 12 15
gross salary=248400.00
gross salary=205500.00
```

18) Predict the output of the following code.

SOURCE CODE:

```
#include<stdio.h>
int main()
{
    char str[] = "peace";
    char *s = str;
    printf("%s\n", s++ +3);
    return 0;
}
```

OUTPUT :ce

19) Predict the output of the following code.

SOURCE CODE:

```
#include<stdio.h>
int main()
{
    int ***r, **q, *p, i=8;
    p = &i;
    q = &p;
    r = &q;
    printf("%d, %d, %d\n", *p, **q,  ***r);
    return 0;
}
```

OUTPUT:8, 8, 8

20) Predict the output of the following code.

SOURCE CODE:

```c
#include<stdio.h>
int main()
{
    int a[3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9,  10, 11, 12 };
    printf("%u, %u, %u\n", a[0]+1,  *(a[0]+1), *(*(a+0)+1));
    return 0;
}
```

OUTPUT: 6487588, 2, 2

PART C

1) Predict the output of the following code.

SOURCE CODE:

```
#include<stdio.h>
int main()
{
      struct a
      {
            float category:5;
            char scheme:4;
      };
      printf("size=%d", sizeof(struct a));
      return 0;
}
```

OUTPUT:

```
Error (invald declaration)
```

2) Predict the output of the following code.

SOURCE CODE:

```
#include<stdio.h>
int main()
{
    struct value
    {
        int bit1:1;
        int bit3:4;
        int bit4:4;
    }bit={1, 2, 13};
    printf("%d, %d, %d\n", bit.bit1,  bit.bit3, bit.bit4);
    return 0;
}
```

OUTPUT:-1, 2, -3

3) Predict the output of the following code.

SOURCE CODE:

```
#include<stdio.h>
int main()
{
    enum days
    {
        MON=-1, TUE, WED=6, THU, FRI,  SAT
    };
    printf("%d, %d, %d, %d, %d, %d\n", MON, TUE, WED, THU, FRI,SAT);
    return 0;
}
```

OUTPUT: -1, 0, 6, 7, 8, 9

4) Identify the error in the following.

SOURCE CODE:

```
#include<stdio.h>
int main()
{
    struct emp
    {
        char name[25];
        intage;
        floatbs;
    };
    struct emp e;
    e.name = "suresh";
    e.age = 25;
    printf("%s %d\n", e.name, e.age);
    return 0;
}
```

OUTPUT:

Error (invalid declaration intage, floatbs should be written as int age, float bs)

5) Predict the output of the following code.

SOURCE CODE:

```c
#include<stdio.h>
struct student {
    char *name;
};
void main() {
    struct student s, m;
    s.name = "st";
    m = s;
    printf("%s%s", s.name, m.name);
}
```

OUTPUT: sts

6) Predict the output of the following code.

SOURCE CODE:

```c
#include<stdio.h>
char s[100];
char *fun(char s[]) {
    static int i = 0;
    if(*s)     {
        fun(s + 1);
        s[i] = *s;
        i++;
    }
    returns;
}
void main() {
    char s[] = "sample code";
    printf("%s", fun(s));
}
```

OUTPUT:

7) Predict the output of the following code.

SOURCE CODE:

```c
#include<stdio.h>
void main() {
    char s1[7] = "1234",*p;
    p = s1 + 2;
```

```
    *p = "\0";
    printf("%s", s1);
}
```

OUTPUT: 12

8) Predict the output of the following code.

SOURCE CODE:

```
#include<stdio.h>
int main() {
    static char *s[] = {"black", "white", "pink", "violet"};
    char **ptr[] = {s+3, s+2, s+1, s}, ***p;
    p = ptr;
    ++p;
    printf("%s", **p+1);
    return 0;
}
```

OUTPUT: ink

9) Predict the output of the following code.

SOURCE CODE:

```
union A {
    char ch;
    int i;
    float f;
}temp;
void main() {
    temp.ch='A';
    temp.i=777;
    temp.f=12345.12345;
    printf("%d", temp.i);
}
```

OUTPUT:

1178657918

10) Predict the output of the following code.

SOURCE CODE:

```c
#include<stdio.h>
void main() {
    struct employee
    {
        unsigned id: 8;
        unsigned sex:1;
        unsigned age:7;
    };
    struct employee emp1={203,1,23};
    printf("%d\t%d\t%d",emp1.id,emp1.sex,emp1.age);
}
```

OUTPUT:

```
203     1       23
```