CHAPTER TWENTY TWO

Miscellaneous Features

[A] What will be the output of the following programs:

(a)
```
# include <stdio.h>
int main( )
{
    enum status { pass, fail, atkt } ;
    enum status  stud1, stud2, stud3 ;
    stud1 = pass ;
    stud2 = fail ;
    stud3 = atkt ;
    printf ( "%d %d %d\n", stud1, stud2, stud3 ) ;
    return 0 ;
}
```

Output:

```
0  1  2
```

(b)
```
# include <stdio.h>
int main( )
{
    printf ( "%f\n", ( float ) ( ( int ) 3.5 / 2 ) ) ;
    return 0 ;
}
```

Output:

1.000000

(c)
```
# include <stdio.h>
int main( )
{
    float i, j ;
    i = ( float ) 3 / 2 ;
    j = i * 3 ;
    printf ( "%d\n", ( int ) j ) ;
    return 0 ;
}
```

Output:

4

[B] Point out the error, if any, in the following programs:

(a)
```
# include <stdio.h>
int main( )
{
    typedef struct patient
    {
        char name[ 20 ] ;
        int age ;
        int systolic_bp ;
        int diastolic_bp ;
    } ptt ;
    ptt p1 = { "anil", 23, 110, 220 } ;
    printf ( "%s %d\n", p1.name, p1.age ) ;
    printf ( "%d %d\n", p1.systolic_bp, p1.diastolic_bp ) ;
    return 0 ;
}
```

No Error

(b)
```
# include <stdio.h>
void show( ) ;
```

```
int main( )
{
    void ( *s )( ) ;
    s = show ;
    ( *s )( ) ;
    s( ) ;
    return 0 ;
}
void show( )
{
    printf ( "don't show off. It won't pay in the long run\n" ) ;
}
```

No Error

(c)
```
# include <stdio.h>
void show ( int, float ) ;
int main( )
{
    void ( *s )( int, float ) ;
    s = show ;
    ( *s )( 10, 3.14 ) ;
    return 0 ;
}
void show ( int i, float f )
{
    printf ( "%d %f\n", i, f ) ;
}
```

Error. Type Mismatch in declaration and definition of **show( )**. While defining **show( )**, return type must be specified as **void**.

[C] Attempt the following:

(a) Create an array of four function pointers. Each pointer should point to a different function. Each of these functions should

receive two integers and return a float. Using a loop call each of these functions using the addresses present in the array.

Program

```
/* To create an array of function pointers */
#include <stdio.h>

float fun1 ( int, int ) ;
float fun2 ( int, int ) ;
float fun3 ( int, int ) ;
float fun4 ( int, int ) ;

float fun1 ( int i, int j )
{
    printf ( "\nIn fun1 %d %d", i, j ) ;
    return ( float )( i / j ) ;
}

float fun2 ( int i, int j )
{
    printf ( "\nIn fun2 %d %d", i, j ) ;
    return ( float )( i / j ) ;
}

float fun3 ( int i, int j )
{
    printf ( "\nIn fun3 %d %d", i, j ) ;
    return ( float )( i / j ) ;
}

float fun4 ( int i, int j )
{
    printf ( "\nIn fun4 %d %d", i, j ) ;
    return ( float )( i / j ) ;
}
```

```
int main( )
{
    float ( *ptr[ 4 ] ) ( int, int ) ;
    float f ;
    int i ;

    ptr [ 0 ] = fun1 ;
    ptr [ 1 ] = fun2 ;
    ptr [ 2 ] = fun3 ;
    ptr [ 3 ] = fun4 ;

    for ( i = 1 ; i < 4 ; i++ )
    {
        f = ( *ptr [ i - 1 ] ) ( 100, i ) ;
        printf ( "%f\n", f ) ;
    }

    return 0 ;
}
```

(b) Write a function that receives variable number of arguments, where the arguments are the coordinates of a point. Based on the number of arguments received, the function should display type of shape like a point, line, triangle, etc. that can be drawn.

Program:

```
/* To recognise type of shape */
# include <stdio.h>

void shape ( int, ... ) ;

int main( )
{
    shape ( 2, 5, 10 ) ;
    shape ( 4, 1, 1, 10, 1 ) ;
    shape ( 6, 15,10, 5, 25, 20, 25 ) ;
```

```
    return 0 ;
}

void shape ( int tot_pt , ... )
{
    switch ( tot_pt )
    {
        case 2 :
            printf ( "Type of shape is point\n" ) ;
            break ;

        case 4 :
            printf ( "Type of shape is line\n" ) ;
            break ;

        case 6 :
            printf ( "Type of shape is triangle\n" ) ;
            break ;
    }
}
```

(c) Write a program, which stores information about a date in a structure containing three members—day, month and year. Using bit fields the day number should get stored in first 5 bits of day, the month number in 4 bits of month and year in 12 bits of year. Write a program to read date of joining of 10 employees and display them in ascending order of year.

Program:

```
/* To store joining dates using bit fields */
# include <stdio.h>

int main( )
{
    struct date
    {
        unsigned day : 5 ;
```

```
    unsigned month : 4 ;
    unsigned year : 12 ;
} ;
struct date dt[ 10 ], temp ;
int i, j, d, m, y ;

printf ( "Enter joining dates (dd-mm-yyyy) of 10 employees\n" ) ;

for ( i = 0 ; i < 10 ; i++ )
{
    scanf ( "%d %d %d", &d, &m, &y ) ;

    if ( ( ( d < 1 ) || ( d > 31 ) ) ||
         ( ( m < 1 ) || ( m > 12 ) ) ||
         ( ( y < 1900 ) || ( y > 2004 ) ) )
    {
        printf ( "Invalid date, enter new date\n" ) ;
        i-- ;
        continue ;
    }

    dt[ i ].day = d ;
    dt[ i ].month = m ;
    dt[ i ].year = y ;
}

for ( i = 0 ; i < 9 ; i++ )
{
    for ( j = i + 1 ; j < 10 ; j++ )
    {
        if ( dt[ j ].year < dt[ i ].year )
        {
            temp = dt[ i ] ;
            dt[ i ] = dt[ j ] ;
            dt[ j ] = temp ;
        }
    }
}
```

```
    for ( i = 0 ; i < 10 ; i++ )
        printf ( "%d %d %d\n", dt[ i ].day, dt[ i ].month, dt[ i ].year ) ;

    return 0 ;
}
```

(d) Write a program to read and store information about insurance policy holder. The information contains details like gender, whether the holder is minor/major, policy name and duration of the policy. Make use of bit-fields to store this information.

Program:

```
/* To store information about insurance policy holder */

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

int main( )
{
    struct policy_holder
    {
        unsigned gender : 1 ;  // 0-Male, 1-Female
        unsigned status : 1 ;  // 0-Minor, 1-Major
        char name[ 20 ] ;
        unsigned dr : 5 ;
    } ;
    struct policy_holder h ;
    int g, s, d ;
    char n[ 20 ] ;

    printf ( "\nEnter gender (0-Male, 1-Female) of the policy holder: " ) ;
    scanf ( "%d", &g ) ;

    printf ( "\nEnter status (0-Minor, 1-Major) of the policy holder: " ) ;
    scanf ( "%d", &s ) ;
```

```
    printf ( "\nEnter name of the policy holder: " ) ;
    scanf ( "%s", n ) ;

    printf ( "\nEnter duration (1 to 25 yrs) of the policy: " ) ;
    scanf ( "%d", &d ) ;

    h.gender = g ;
    h.status = s ;
    strcpy ( h.name, n ) ;
    h.dr = d ;

    printf ( "Name: %s\n", h.name ) ;
    printf ( "Gender: %s\n", h.gender == 0 ? "Male" : "Female" ) ;
    printf ( "Status: %s\n", h.status == 0 ? "Minor" : "Major" ) ;
    printf ( "Duration %d\n", h.dr ) ;

    return 0 ;
}
```