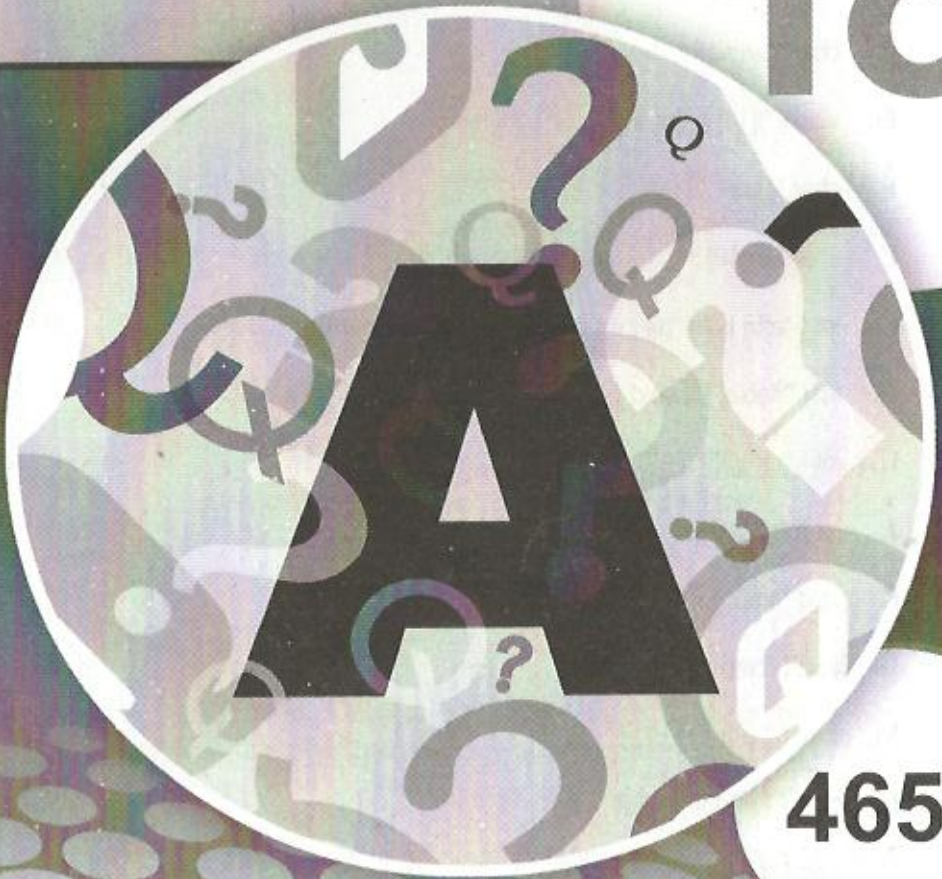
18 Complicated Declarations



Question *18.1*

What do the following declarations signify?

A. `int *f( ) ;`

B. `int ( *pf )( ) ;`

C. `char **argv ;`

D. `void ( *f[10] ) ( int, int ) ;`

E. `char *scr ;`

F. `char *arr[10] ;`

G. `int ( *a[5] )( int *p ) ;`

H. `int ( *ftable[ ] )( void ) = { fadd, fsub, fmul, fdiv } ;`

I. `int ( *ptr )[30] ;`

J. `int *ptr[30] ;`

K. `void *cmp( ) ;`

L. `void ( *cmp )( ) ;`

M. `char ( * ( *f( ) ) [ ] ) ( ) ;`

N. `char ( * ( *x[3] )( ) )[5] ;`

O. `void ( *f )( int, void ( * )( ) ) ;`

P. `int ** ( *f )( int **, int ** ( * ) ( int **, int ** ) ) ;`

Q. `void ( *f )( void ( * )( int *, void **), int ( * )(void **, int *)) ;`

R. `typedef void ( *pfun ) ( int , float ) ;`

S. `char far *scr1, *scr2 ;`

T. `char far * far *ptr ;`

U. `char far * near *ptr ;`

V. `char far * huge *ptr ;`

Answer

A. *f* is a function returning pointer to an *int*.

B. *pf* is a pointer to function which returns an *int*.

C. *argv* is a pointer to a *char* pointer.

D. *f* is an array of 10 function pointers, where each pointer points to a function that receives two *int*s and returns nothing.

E. *scr* is a pointer to a pointer to a *char*.

F. *arr* is an array of 10 character pointers.

G. *a* is an array of 5 function pointers. Each of these pointers point to a function that receives an *int* pointer and returns an *int*.

H. *ftable* is an array of 4 function pointers which points to the functions *fadd( )*, *fsub( )*, etc. Each of these functions accepts nothing and returns an *int*.

I. *ptr* is a pointer to an array of 30 integers.

Question *18.1*

What do the following declarations signify?

A. `int *f( ) ;`

B. `int ( *pf )( ) ;`

C. `char **argv ;`

D. `void ( *f[10] ) ( int, int ) ;`

E. `char *scr ;`

F. `char *arr[10] ;`

G. `int ( *a[5] )( int *p ) ;`

H. `int ( *ftable[ ] )( void ) = { fadd, fsub, fmul, fdiv } ;`

I. `int ( *ptr )[30] ;`

J. `int *ptr[30] ;`

K. `void *cmp( ) ;`

L. `void ( *cmp )( ) ;`

M. `char ( * ( *f( ) ) [ ] ) ( ) ;`

N. `char ( * ( *x[3] )( ) )[5] ;`

O. `void ( *f )( int, void ( * )( ) ) ;`

P. `int ** ( *f )( int **, int ** ( * ) ( int **, int ** ) ) ;`

Q. `void ( *f )( void ( * )( int *, void **), int ( * )(void **, int *)) ;`

R. `typedef void ( *pfun ) ( int , float ) ;`

S. `char far *scr1, *scr2 ;`

T. `char far * far *ptr ;`

U. `char far * near *ptr ;`

V. `char far * huge *ptr ;`

Answer

A. *f* is a function returning pointer to an *int*.

B. *pf* is a pointer to function which returns an *int*.

C. *argv* is a pointer to a *char* pointer.

D. *f* is an array of 10 function pointers, where each pointer points to a function that receives two *int*s and returns nothing.

E. *scr* is a pointer to a pointer to a *char*.

F. *arr* is an array of 10 character pointers.

G. *a* is an array of 5 function pointers. Each of these pointers point to a function that receives an *int* pointer and returns an *int*.

H. *ftable* is an array of 4 function pointers which points to the functions *fadd( )*, *fsub( )*, etc. Each of these functions accepts nothing and returns an *int*.

I. *ptr* is a pointer to an array of 30 integers.

J. *ptr* is an array of 30 pointers to integers.

K. *cmp* is a function that returns a *void* pointer.

L. *cmp* is a pointer to function which returns a *void*.

M. *f* is a function that returns a pointer to an array of pointers to functions that return a *char*.

N. *x* is an array of 3 pointers to functions that return a pointer to an array of 5 *char*s.

O. *f* is a pointer to a function which returns nothing and receives as its parameter an integer and a pointer to a function which receives nothing and returns nothing.

P. *f* is a pointer to a function which returns a pointer to an int pointer and receives two arguments—a pointer to an *int* pointer and a function pointer which points to a function which receives two pointers to *int* pointers and returns a pointer to an *int* pointer.

Q. *f* is a pointer to a function which returns nothing and receives two arguments, both function pointers. The first function pointer points to a function which returns nothing but receives two arguments—an *int* pointer and a pointer to a *void* pointer; the second function pointer points to a function which returns an *int* pointer and receives a pointer to a *void* pointer and an *int* pointer.

R. *pfun* is the new name for a data type which is pointer to function that accepts two arguments and returns nothing. Once this type is defined we can create variables of this type through statements like *pfun f1*.

S. *scr1* is a *far* pointer to a *char*, whereas *scr2* is a *near* pointer to a *char*.

T. *ptr* is a *far* pointer to a *far* pointer to a *char*, or in easier words, *ptr* contains a *far* address of a *far* pointer to a *char*.

U. *ptr* is a *huge* pointer to a *far* pointer to a *char*, or in easier words, *ptr* contains a *huge* address of a *far* pointer to a *char*.

V. *ptr* is a *near* pointer to a *far* pointer to a *char*, or in easier words, *ptr* contains a *near* address of a *far* pointer to a *char*.

Question *18.2*

What will be the output of the following program?

```
#include <stdio.h>
int main( )
{
    char near * near *ptr1 ;
    char near * far  *ptr2 ;
    char near * huge *ptr3 ;
    printf ( "%d %d %d\n", sizeof ( ptr1 ), sizeof ( ptr2 ), sizeof ( ptr3 ) ) ;
    return 0 ;
}
```

Answer

2 4 4

Note: *near*, *far* and *huge* pointers work only in TC/TC++ under DOS. In Visual Studio under Windows and gcc under Linux there are no *near*, *far* and *huge* pointers.

Question 18.3

What will be the output of the following program?

```
#include <stdio.h>
int main( )
{
    char far * near *ptr1 ;
    char far * far  *ptr2 ;
    char far * huge *ptr3 ;
    printf ( "%d %d %d\n", sizeof ( ptr1 ), sizeof ( ptr2 ), sizeof ( ptr3 ) ) ;
    return 0 ;
}
```

Answer

2 4 4

Note: *near*, *far* and *huge* pointers work only in TC/TC++ under DOS. In Visual Studio under Windows and gcc under Linux there are no *near*, *far* and *huge* pointers.

Question 18.4

What will be the output of the following program?

```
#include <stdio.h>
int main( )
{
    char huge * near *ptr1 ;
    char huge * far  *ptr2 ;
    char huge * huge *ptr3 ;
    printf ( "%d %d %d\n", sizeof ( ptr1 ), sizeof ( ptr2 ), sizeof ( ptr3 ) ) ;
    return 0 ;
}
```


Answer

2 4 4

Note: *near*, *far* and *huge* pointers work only in TC/TC++ under DOS. In Visual Studio under Windows and gcc under Linux there are no *near*, *far* and *huge* pointers.

Question 18.5

What will be the output of the following program?

```
#include <stdio.h>
int main( )
{
    char huge * near * far  *ptr1 ;
    char near * far  * huge *ptr2 ;
    char far  * huge * near *ptr3 ;
    printf ( "%d %d %d\n", sizeof ( ptr1 ), sizeof ( ptr2 ), sizeof ( ptr3 ) ) ;
    return 0 ;
}
```

Answer

4 4 2

Note: *near*, *far* and *huge* pointers work only in TC/TC++ under DOS. In Visual Studio under Windows and gcc under Linux there are no *near*, *far* and *huge* pointers.

Question 18.6

What will be the output of the following program?

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

```
{
    char huge * near * far  *ptr1 ;
    char near * far  * huge *ptr2 ;
    char far  * huge * near *ptr3 ;
    printf ( "%d %d %d\n", sizeof ( ptr1 ), sizeof ( *ptr2 ), sizeof ( **ptr3 ) ) ;
    return 0 ;
}
```

Answer

4 4 4

Note: *near*, *far* and *huge* pointers work only in TC/TC++ under DOS. In Visual Studio under Windows and gcc under Linux there are no *near*, *far* and *huge* pointers.

Question *18.7*

What will be the output of the following program?

```
#include <stdio.h>
int main( )
{
    char huge * near * far  *ptr1 ;
    char near * far  * huge *ptr2 ;
    char far  * huge * near *ptr3 ;
    printf ( "%d %d %d\n", sizeof ( *ptr1 ), sizeof ( **ptr2 ), sizeof ( ptr3 ) ) ;
    return 0 ;
}
```

Answer

2 2 2

Note: *near*, *far* and *huge* pointers work only in TC/TC++ under DOS. In Visual Studio under Windows and gcc under Linux there are no *near*, *far* and *huge* pointers.

Question *18.8*

What will be the output of the following program?

```
#include <stdio.h>
int main( )
{
    char huge * near * far  *ptr1 ;
    char near * far  * huge *ptr2 ;
    char far  * huge * near *ptr3 ;
    printf ( "%d %d %d\n", sizeof ( **ptr1 ), sizeof ( ptr2 ), sizeof ( *ptr3 ) ) ;
    return 0 ;
}
```

Answer

4 4 4

Note: *near*, *far* and *huge* pointers work only in TC/TC++ under DOS. In Visual Studio under Windows and gcc under Linux there are no *near*, *far* and *huge* pointers.

Question *18.9*

Are the following two declarations same? [Yes/No]

```
char far * far *scr ;
char far far ** scr ;
```

Answer

No

Question *18.10*

How will you declare the following?

A. An array of three pointers to chars.
B. An array of three *char* pointers.
C. A pointer to an array of three *chars*.
D. A pointer to a function which receives an *int* pointer and returns a *float* pointer.
E. A pointer to a function which receives nothing and returns nothing.

Answer

A. char *ptr[3] ;
B. char *ptr[3] ;
C. char (*ptr)[3] ;
D. float * (*ptr)(int *) ;
E. void (*ptr)( ) ;

Question *18.11*

Can you write a program which will implement the following declaration?

```
void ( *f )( int, void ( * )( ) ) ;
```

Answer

```
#include <stdio.h>
int main( )
{
    void ( *f )( int, void ( * )( ) ) ;
    void fun ( int, void ( * )( ) ) ;
    void fun1( ) ;
    void ( *p )( ) ;
```



```
    f = fun ;
    p = fun1 ;
    ( *f )( 23, p ) ;
    return 0 ;
}
void fun ( int i, void ( *q )( ) )
{
    printf ( "Hello\n" ) ;
}
void fun1( )
{
    ;
}
```

Question *18.12*

Point out the error, if any, in the following code.

```
#include <stdio.h>

void display ( int ( *ff ) ( ) ) ;
int main( )
{
    int show( ) ;
    int ( *f )( ) ;
    f = show ;
    display ( f ) ;
    return 0 ;
}
void display ( int ( *ff ) ( ) )
{
    ( *ff )( ) ;
}
int show( )
{
    printf ( "On the rebound ....\n" ) ;
    return 0 ;
```

```
}
```

Answer

No error because we pass address of *show( )* function to *display( )* function and in *display( )* we call *show( )* through its address.

Question *18.13*

What will be the output of the following program?

```
#include <stdio.h>
int main( )
{
    struct s1
    {
        char *z ;
        int i ;
        struct s1 *p ;
    } ;
    static struct s1 a[ ] = { { "Nagpur", 1, a + 1 }, { "Raipur", 2, a + 2 },
                             { "Jabalpur", 3, a } } ;
    struct s1 *ptr = a ;
    printf ( "%s\n", ++( ptr -> z ) ) ;
    printf ( "%s\n", a [ ( ++ptr ) -> i ].z ) ;
    printf ( "%s\n", a[--( ptr -> p -> i )].z ) ;
    return 0 ;
}
```

Answer

```
agpur
Jabalpur
Jabalpur
```


Question *18.14*

What will be the output of the following program?

```
#include <stdio.h>
typedef unsigned long int uli ;
typedef uli u ;
int main( )
{
    uli a ;
    u b = -1 ;
    a = -1 ;
    printf ( "%lu %lu\n", a,b ) ;
    return 0 ;
}
```

Answer

4294967295 4294967295

Question *18.15*

What will be the output of the following program?

```
#include <stdio.h>
double i ;
int main( )
{
    ( int )( float )( char ) i ;
    printf ( "%d\n", sizeof ( i ) ) ;
    return 0 ;
}
```

A. 1
B. 2
C. 4

D. 8

Answer

D

Question *18.16*

What will be the output of the following program?

```
#include <stdio.h>
double i ;
int main( )
{
    ( int )( float )( char ) i ;
    printf ( "%d\n", sizeof ( ( int )( float )( char ) i ) ) ;
    return 0 ;
}
```

A. 1
B. 2
C. 4
D. 8

Answer

B in case of TC/TC++, C in case of Visual Studio, gcc.

Question *18.17*

What will be the output of the following program?

```
#include <stdio.h>
typedef void v ;
typedef int i ;
int main( )
{
    v fun ( i, i ) ;
    fun ( 2, 3 ) ;
    return 0 ;
}
v fun( i a, i b )
{
    i s = 2 ;
    float i ;
    printf ( "%d ", sizeof ( i ) ) ;
    printf ( "%d\n", a * b * s ) ;
}
```


Answer

4 12

Question *18.18*

What's wrong with this declaration?

```
#include <stdlib.h>
int main( )
{
    static char *p = ( char* ) malloc( 10 ) ;
    return 0 ;
}
```

Answer

No error.

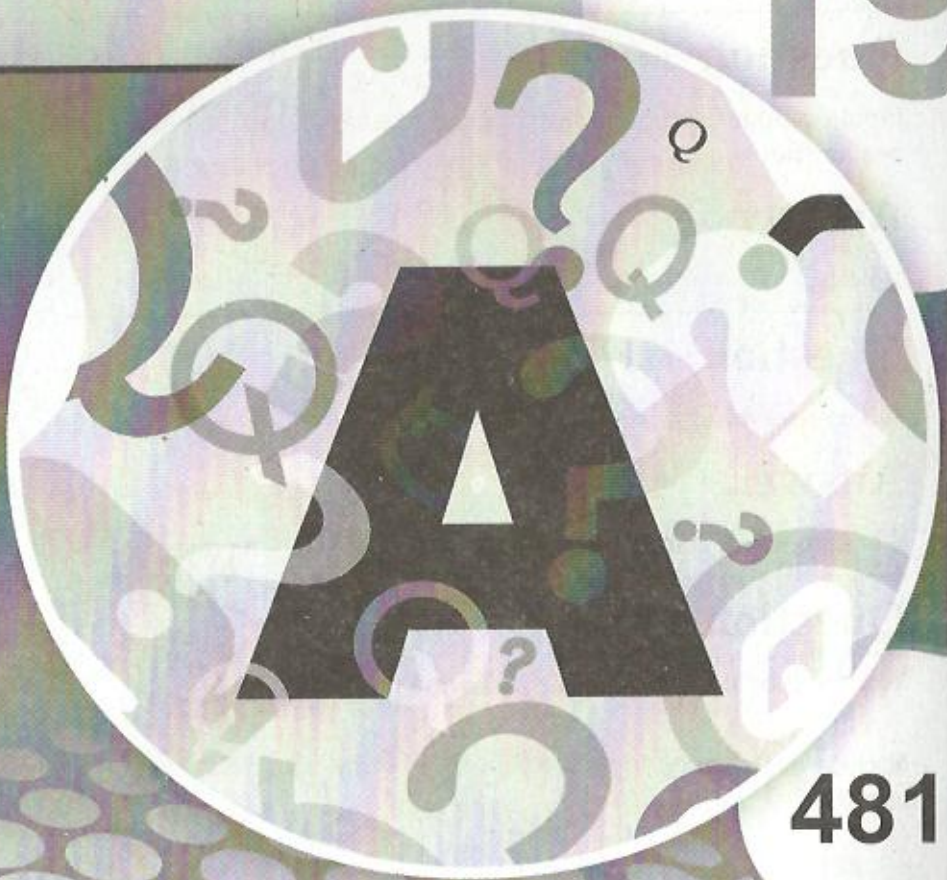
Question *18.19*

Point out the error, if any, in the following code.

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

Library Functions

19



481