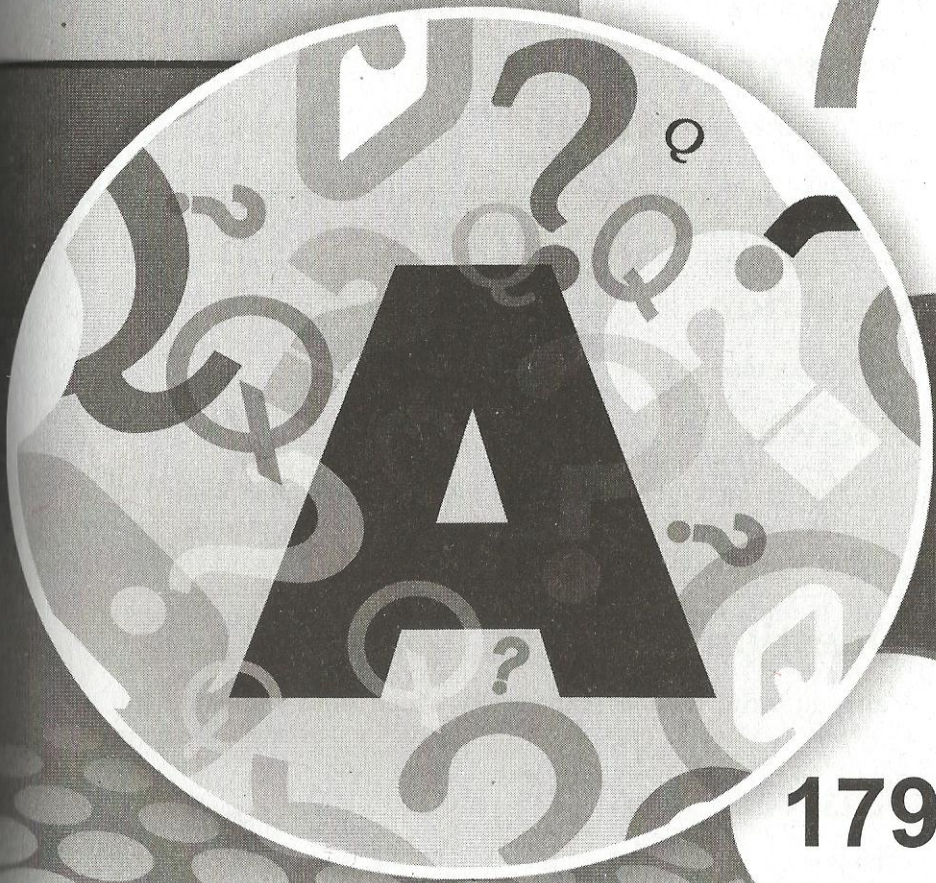
7 Pointers



Question 7.1

Can you combine the following two statements into one?

```
char *p ;
p = ( char * ) malloc ( 100 ) ;
```

Answer

```
char *p = ( char * ) malloc ( 100 ) ;
```

Note that the typecasting operation can be dropped completely if this program is built using gcc compiler.

Question 7.2

Are the expressions **ptr++* and *++*ptr* same?

Answer

No. **ptr++* increments the pointer and not the value pointed by it, whereas *++*ptr* increments the value being pointed to by *ptr*. *++*ptr* works like **ptr = *ptr + 1*.

Question 7.3

Can you write another expression which does the same job as *++*ptr*?

Answer

```
( *ptr )++
```

Question 7.4

What would be the equivalent pointer expression for referring the array element *a[i][j][k][l]*?

Answer

```
*( *( *( *( a + i ) + j ) + k ) + l )
```

Question 7.5

If size of an integer is 4 bytes what will be the output of the following program?

```
#include <stdio.h>
int main( )
{
    int arr[ ] = { 12, 13, 14, 15, 16 } ;
    printf ( "%d %d %d\n", sizeof ( arr ), sizeof ( *arr ), sizeof ( arr[0] ) ) ;
    return 0 ;
}
```

Answer

20 4 4

Question 7.6

What will be the output of the following program assuming that the array begins at location 1002 and size of an integer is 4 bytes?

```
#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 ;
}
```

```
}
```

Answer

1006 2 2

Question 7.7

Will the following program compile?

```
#include <stdio.h>
int main( )
{
    char str[5]  = "fast enough" ;
    return 0 ;
}
```

Answer

Yes. The compiler never detects the error if bounds of an array are exceeded.

Question 7.8

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    int arr[3] = { 2, 3, 4 } ;
    char *p ;
    p = ( char * ) arr ;
    printf ( "%d ", *p ) ;
    p = p + 1 ;
    printf ( "%d \n", *p ) ;
```



```
    return 0 ;
}
```

A. 2 3
B. 2 0
C. 1 0
D. Garbage values

Answer

B

Question 7.9

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    int arr[2][2][2] = { 10, 2, 3, 4, 5, 6, 7, 8 } ;
    int *p, *q ;
    p = &arr[1][1][1] ;
    q = ( int * ) arr ;
    printf ( "%d %d\n", *p, *q ) ;
    return 0 ;
}
```

A. 8 10
B. 10 2
C. 8 1
D. Garbage values

Answer

A

Question 7.10

What will be the output of the following program assuming that the array begins at location 1002?

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

Answer

1002 1002 1002 1

Question 7.11

In the following program how will you print 50 using *p*?


```
#include <stdio.h>
int main( )
{
    int a[ ] = { 10, 20, 30, 40, 50 } ;
    char *p ;
    p = ( char * ) a ;
    return 0 ;
}
```

Answer

```
printf ( "%d\n", * ( ( int * ) p + 4 ) ) ;
```

Question 7.12

Where can one think of using pointers?

Answer

At lot of places, some of which are:

- Accessing array or string elements
- In passing big objects like arrays, strings and structures to functions
- Dynamic memory allocation
- Call by reference
- Implementing linked lists, trees, graphs and many other data structures

Question 7.13

In the following program add a statement in the function *fun( )* such that address of *a* gets stored in *j*.

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

```
int main( )
{
    int *j ;
    void fun ( int ** ) ;
    fun ( &j ) ;
    return 0 ;
}
void fun ( int **k )
{
    int a = 10 ;
    /* add statement here */
}
```

Answer

```
*k = &a ;
```

Question 7.14

How will you declare an array of three function pointers where each function receives two *ints* and returns a *float*?

Answer

```
float ( *arr[3] ) ( int, int ) ;
```

Question 7.15

Will the following program report an error on compilation? [Yes/No]

```
#include <stdio.h>
int main( )
{
    float i = 10, *j ;
    void *k ;
    k = &i ;
    j = k ;
    printf ( "%f\n", *j ) ;
    return 0 ;
}
```


Answer

No error will be reported if the program is compiled using gcc. In gcc no typecasting is required while assigning the value to and from *k* because conversions are applied automatically when other pointer types are assigned to and from *void**.

However, when compiled using Visual Studio compiler, it reports errors as it demands typecasting.

Question 7.16

Will the following program compile?

```
#include <stdio.h>
int main( )
{
    int a = 10, *j ;
    void *k ;
    j = k = &a ;
    j++ ;
    k++ ;
    printf ( "%u %u\n", j, k ) ;
    return 0 ;
}
```

Answer

No. An error would be reported in the statement *k++* since arithmetic on *void* pointers is not permitted unless the *void* pointer is appropriately typecasted.

Question 7.17

What will be the output of the following program?

```
#include <stdio.h>
int main( )
{
    printf ( "%c\n", 7[ "Sundaram" ] ) ;
    return 0 ;
}
```

Answer

It would print *m* of *Sundaram*.

Question 7.18

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    char *p ;
    p = "hello" ;
    printf ( "%s \n", *&*&p ) ;
    return 0 ;
}
```

A. llo
B. hello
C. ello
D. h

Answer

B

Question 7.19

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    void *vp ;
    char ch = 74, *cp = "JACK" ;
    int j = 65 ;
    vp = &ch ;
    printf ( "%c", * ( char * ) vp ) ;
    vp = &j ;
    printf ( "%c", * ( int * ) vp ) ;
    vp = cp ;
    printf ( "%s\n", ( char * ) vp + 2 ) ;
    return 0 ;
}
```

A. JCK
B. J65K
C. JAK
D. JACK

Answer

D

Question 7.20

Which of the following is the correct output for the program given below?

```
#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 \n", **p + 1 ) ;
    return 0 ;
}
```

A. ink
B. ack
C. ite
D. let

Answer

A

Question 7.21

Which of the following is the correct output for the program given below?

```
#include <string.h>
#include <stdio.h>
int main( )
{
    int i, n ;
    char *x = "Alice" ;
    n = strlen ( x ) ;
    for ( i = 0 ; i <= n ; i++ )
    {
        printf ( "%s ", x ) ;
        x++ ;
    }
    printf ( "\n", x ) ;

    return 0 ;
}
```

A. No output
B. ecilA
C. Alice lice ice ce e
D. lice ice ce e

Answer

C

Question 7.22

Which of the following is the correct output for the program given below, if an integer is 4 bytes long and *r* begins in memory at address 4000?

```
#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 ;
}
```

A. 8 8 8
B. 4000 4002 4004
C. 4000 4004 4008
D. 4000 4008 40016

Answer

A

Question 7.23

Which of the following is the correct output for the program given below?

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

A. Error: Cannot convert from void ** to int **
B. Garbage value
C. 0
D. No output

Answer

C

Question 7.24

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    char *str ;
    str = "%d\n" ;
    str++ ;
    str++ ;
    printf ( str - 2, 300 ) ;
    return 0 ;
}
```

A. No output
B. 30
C. 3
D. 300

Answer

D

Question 7.25

Which of the following is the correct output for the program given below?

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

```
{
    char *str ;
    str = "%s" ;
    printf ( str, "K\n" ) ;
    return 0 ;
}
```

A. Error
B. No output
C. K
D. %s

Answer

C

Question *7.26*

In the following program add a statement in the function *fact( )* such that the factorial gets stored in *j*.

```
#include <stdio.h>
void fact ( int * ) ;
int main( )
{
    int i = 5, j ;
    fact ( &i ) ;
    printf ( "%d\n", i ) ;
    return 0 ;
}
void fact ( int *j )
{
    static int s = 1 ;
    if ( *j != 0 )
    {
        s = s * *j ;
        *j = *j - 1 ;
```



```
        fact ( j ) ;
        /* add statement here */
    }
}
```

Answer

```
*j = s ;
```

Question *7.27*

Which of the following is the correct output for the program given below?

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

A. 30
B. 27
C. 9
D. 3

Answer

A

Question 7.28

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
void change ( int *, int ) ;

int main( )
{
    int i, a[ ] = { 2, 4, 6, 8, 10 } ;
    change ( a, 5 ) ;
    for ( i = 0 ; i <= 4 ; i++ )
        printf ( "%d, ", a[ i ] ) ;
    printf ( "\n" ) ;
    return 0 ;
}
void change ( int *b, int n )
{
    int i ;
    for ( i = 0 ; i < n ; i++ )
        * ( b + i ) = * ( b + i ) + 5 ;
}
```

A. 7, 9, 11, 13, 15
B. 2, 4, 6, 8, 10
C. 2 4 6 8 10
D. 3, 1, -1, -3, -5

Answer

A

Question 7.29

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int * check ( static int , static int  ) ;
int main( )
{
    int *c ;
    c = check ( 10, 20 ) ;
    printf ( "%d\n", *c ) ;
    return 0 ;
}
int * check ( static int i, static int j )
{
    int *p, *q ;
    p = &i ;
    q = &j ;
    if ( i >= 45 )
        return ( p ) ;
    else
        return ( q ) ;
}
```

A. 10
B. 20
C. Error: 'Non portable pointer conversion'
D. Error: 'Cannot use static for function parameters'

Answer

B

Question 7.30

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    int x = 30, *y, *z ;
    y = &x ; /* suppose address of x is 500 and integer is 4 byte wide */
    z = y ;
    *y++ = *z++ ;
    x++ ;
    printf ( "x = %d y = %d z = %d\n", x, y, z ) ;
    return 0 ;
}
```

A. 31 502 502
B. 31 500 500
C. 31 498 498
D. 31 504 504

Answer

D

Question 7.31

Which of the following error will be reported on compiling the program given below?

```
#include <stdio.h>
int main( )
{
    int a[ ] = { 10, 20, 30, 40, 50 } ;
    int j ;
    for ( j = 0 ; j < 5 ; j++ )
    {
        printf ( "%d\n", *a ) ;
        a++ ;
    }
    return 0 ;
}
```

A. Error: 'Declaration syntax'
B. Error: 'Expression syntax'
C. Error: 'Lvalue required'
D. Error: 'Rvalue required'

Answer

C

Question 7.32

Which of the following statement is correct about *k* used in the statement given below?

```
char ****k ;
```

A. *k* is Pointer to a pointer to a pointer to a *char*.
B. *k* is Pointer to a pointer to a pointer to a pointer to a *char*.
C. *k* is Pointer to a *char* pointer.
D. *k* is Pointer to a pointer to a *char*.

Answer

B

Question 7.33

Which of the following is the correct output for the program given below?

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

What will be the output of the above code?

A. peace
B. eace
C. ace
D. ce

Answer

D

Question *7.34*

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    char str1[ ] = "Bombay" ;
    char str2[ ] = "Pune" ;
    char *s1= str1, *s2 = str2 ;
    while ( *s1++ = *s2++ ) ;
        printf ( "%s", str1 ) ;
    printf ( "\n" ) ;
    return 0 ;
}
```


A. BombayPune
B. Pune
C. Bombay.
D. (null)

Answer

B

Question *7.35*

Which of the following statements correctly declare a function that receives a pointer to a pointer to a pointer to a *float* and returns a pointer to a pointer to a pointer to a pointer to a *float*?

A. float ** fun (float ***) ;
B. float * fun (float **) ;
C. float fun (float ***) ;
D. float **** fun (float ***) ;

Answer

D

Question *7.36*

Which statement will you add to the following program to ensure that the program outputs "Mumbadevi" on execution?

```
#include <stdio.h>
int main( )
{
    char s[ ] = "Mumbadevi" ;
    char t[25] ;
    char *ps, *pt ;
```

```
    ps = s ;
    pt = t ;
    while ( *ps )
        *pt++ = *ps ++ ;
    /* add suitable statement here */
    printf ( "%s\n", t ) ;
    return 0 ;
}
```

Answer

```
*pt = '\0' ;
```

Question *7.37*

Which of the following statements is correct about the program given below?

```
#include <stdio.h>
int main( )
{
    int arr[3][3] = { 1, 2, 3, 4 } ;
    printf ( "%d\n", * ( * ( * ( arr ) ) ) ) ;
    return 0 ;
}
```

A. It will output a garbage value.
B. It will output a value 1.
C. It will output a value 3.
D. It will report an error: 'Invalid indirection'.

Answer

D

Question *7.38*

If a variable is a pointer to a structure, then which of the following operator is used to access data members of the structure through the pointer variable?

A. '.'
B. '&'
C. '*'
D. '->'

Answer

D

Question *7.39*

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    char str[20] = "Hello" ;
    char * const p = str ;
    *p = 'M' ;
    printf ( "%s\n", str ) ;
    return 0 ;
}
```

A. Mello
B. Hello
C. HMello
D. MHello

Answer

A

Question 7.40

The operator used to get value at address stored in a pointer variable is

A. *
B. &
C. &&
D. ||

Answer

A

Question 7.41

A pointer is

A. A keyword used to create variables.
B. A variable that stores address of an instruction.
C. A variable that stores address of another variable.
D. All of the above

Answer

C

Question 7.42

Which of the following statement is true about the program given below?


```
#include <stdio.h>
int main( )
{
    float a = 3.14 ;
    char *j ;
    j = ( char * ) &a ;
    printf ( "%d\n", *j ) ;
    return 0 ;
}
```

A. It will print ASCII value of the binary number present in the first byte of the *float* variable *a*.
B. It will print character equivalent of the binary number present in the first byte of the *float* variable *a*.
C. It will print 3.
D. It will print a garbage value

Answer

A

Question 7.43

Which of the following statement is correct about the program given below?

```
#include <stdio.h>
int main( )
{
    int i = 10 ;
    int *j = &i ;
    return 0 ;
}
```

A. *j* and *i* are pointers to an *int*.
B. *i* is a pointer to an *int* and stores address of *j*.

C. *j* is a pointer to an *int* and stores address of *i*.
D. *j* is a pointer to a pointer to an *int* and stores address of *i*.

Answer

C

Question *7.44*

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int power ( int ** ) ;
int  main( )
{
    int a = 5, *aa ;  /* suppose the address of the variable a is 1000 */
    aa = &a ;
    a = power ( &aa ) ;
    printf ( "%d \n", a ) ;
    return 0 ;
}
int power ( int **ptr )
{
    int b ;
    b = **ptr ***ptr ;
    return ( b ) ;
}
```

A. 5
B. 25
C. 125
D. Garbage value

Answer

B

Question *7.45*

Is the NULL pointer same as an uninitialised pointer? [Yes/No]

Answer

No

Question *7.46*

In which header file is the NULL macro defined.

Answer

In files "stdio.h" and "stddef.h".

Question *7.47*

Is there any difference between the following two statements?

```
char *p = 0 ;
char *t = NULL ;
```

Answer

No. NULL is *#defined* as 0 in the 'stdio.h' file. Thus, both *p* and *t* are null pointers.

Question *7.48*

What is (void *) 0?

Answer

One of the representation for Null Pointer.

Question *7.49*

Is this a correct way for null pointer assignment?

```
int i = 0 ;
char *q = ( char * ) i ;
```

Answer

No. The correct way is shown below:

```
char *q = 0 ;
```

Or

```
char *q = ( char * ) 0 ;
```

Question *7.50*

What is a null pointer?

Answer

For each pointer type (like say a *char* pointer) C defines a special pointer value, which is guaranteed not to point to any object or function of that type. Usually, the null pointer constant used for representing a null pointer is the integer 0.

Question *7.51*

What's the difference between a null pointer, a NULL macro, the ASCII NUL character and a null string?

Answer

A null pointer is a pointer, which doesn't point anywhere.

A NULL macro is used to represent the null pointer in source code. It has a value 0 associated with it.

The ASCII NUL character has all its bits as 0 but doesn't have any relationship with the null pointer.

The null string is just another name for an empty string "".

Question *7.52*

What will be the output of the following program if the size of the pointer is considered as 4 bytes?

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

Answer

4 1

Question *7.53*

Point out the error, if any, in the following program

```
#include <stdio.h>
int main( )
{
    int *x ;
    *x = 100 ;
```

```
    return 0 ;
}
```

Answer

There isn't any syntax error in this code. However, since pointer *x* is not initialized with any memory address it will hold some unpredictable address at initialization. It is a bad practice to use a variable (*x* in this case) without initializing it.

Question *7.54*

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

```
*p = ( char * ) malloc ( 100 ) ;
*p = 'y' ;
```

Answer

The base address of memory allocated by *malloc* should get collected in *p*. Here it will get collected in some unknown address as *p* is not initialized. Thus, the correct statement will be

```
p = ( char * ) malloc ( 100 ) ;
```

Question *7.55*

How many bytes are occupied by *near*, *far* and *huge* pointers?

Answer

A *near* pointer is 2 bytes long whereas a *far* and a *huge* pointer are 4 bytes long. Note that *near*, *far* and *huge* pointers exist only under DOS. Under Windows and Linux there is no distinction like *near*, *far* or *huge*. In Windows and Linux every pointer is 4 bytes long.

Question *7.56*

Are the three declarations *char **apple*, *char *apple[ ]*, and *char apple[ ][ ]* same? [Yes/No]

Answer

No

Question *7.57*

Will the following program give any warning on compilation?

```
#include <stdio.h>
void main( )
{
    int *p1, i = 25 ;
    void *p2 ;
    p1 = &i ;
    p2 = &i ;
    p1 = p2 ;
    p2 = p1 ;
    return 0 ;
}
```

Answer

In Visual Studio a warning would be reported for the statement *p1 = p2 ;* saying 'a value of type *void* * cannot be assigned to an entity of type *int* *. In gcc no such warning would be reported.

Question *7.58*

Since size of pointer as well as size of standard data types varies from one compiler to another, how should I write programs that work similarly with all compilers?

Answer

To keep your programs portable across all compilers avoid statements where you make an assumption about the size of pointer or size of a standard data type. Instead use the sizeof operator to use the correct size of a pointer or a standard data type. This is shown below with an example:

```
int *p ;
p = ( int * ) malloc ( 10 * 4 ) ;
```

Here we have reserved space for 10 integers, assuming the size of an integer to be 4 bytes. Instead a better way would have been:

```
int *p ;
p = ( int * ) malloc ( 10 * sizeof ( int ) ) ;
```