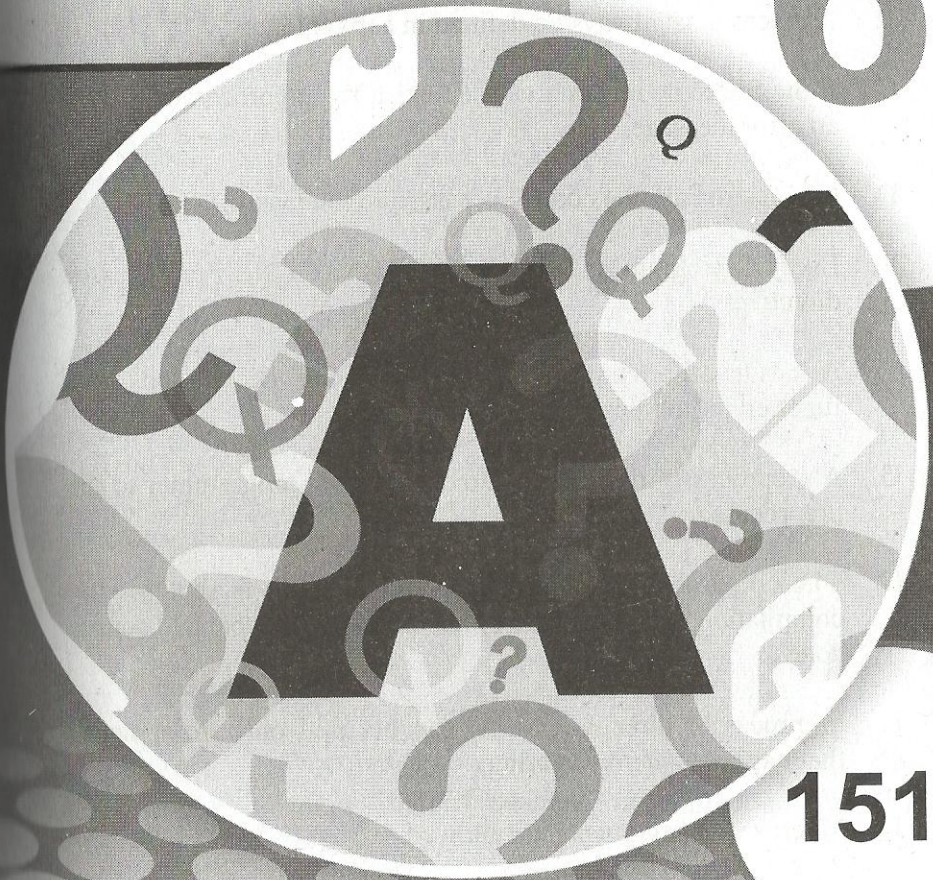
The C Preprocessor



Question 6.1

State True or False:

A. A preprocessor directive is a message from compiler to the linker.

B. A macro must always be defined in capital letters.

C. If the file to be included doesn't exist, the preprocessor flashes an error message.

D. In a macro call, the control is passed to the macro.

E. Every C program will contain AT LEAST ONE Preprocessor directive.

F. A header file contains macros, structure declarations and function prototypes.

G. A preprocessor directive is a message from programmer to the preprocessor.

H. Once preprocessing is over and the program is sent for the compilation the macros are removed from the expanded source code.

I. Preprocessor directive *#undef* can be used only on a macro that has been *#define*d earlier.

J. Macros with arguments are allowed.

K. Macros have a local scope.

L. Preprocessor directive *#ifdef .. #else .. #endif* is used for conditional compilation.

M. There exist ways to prevent the same file from getting *#include*d twice in the same program.

N. Macro calls and function calls work exactly similarly.

O. The preprocessor can trap simple errors like missing declarations, nested comments or mismatch of braces.

Answer

A. False
B. False
C. True
D. False
E. False
F. True
G. True
H. True
I. True
J. True
K. False
L. True
M. True
N. False
O. False

Question 6.2

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

```
#include <stdio.h>
#define SQR( x ) ( x * x )
int main( )
{
```

```
    int a, b = 3 ;
    a = SQR ( b + 2 ) ;
    printf ( "%d\n", a ) ;
    return 0 ;
}
```

A. 25
B. 11
C. Error
D. Garbage value

Answer

B. Because, on preprocessing the expression becomes *a = (3 + 2 * 3 + 2)*

Question *6.3*

How will you define the SQR macro in 6.2 above such that it gives the result of *a* as 25?

Answer

```
#define SQR(x) ( (x) * (x) )
```

Question *6.4*

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

```
#include <stdio.h>
#define SQUARE(x) x * x
int main( )
{
    float s = 10, u = 30, t = 2, a ;
    a = 2 * ( s - u * t ) / SQUARE ( t ) ;
```



```
    printf ( "Result: %f\n" , a ) ;
    return 0 ;
}
```

A. Result: -100.000000
B. Result: -25.000000
C. Result: 0.000000
D. Result: 100.000000

Answer

A

Question *6.5*

What will be the output of the following program?

```
#include <stdio.h>
#define CUBE(x) ( x * x * x )
int main( )
{
    int a, b = 3 ;
    a = CUBE ( b++ ) ;
    printf ( "%d %d\n", a, b ) ;
    return 0 ;
}
```

Answer

27 6. Though some compilers may give this as the answer, according to the ANSI C standard the expression *b++ * b++ * b++* is undefined. Refer Chapter 3 for more details on this.

Question 6.6

What will the SWAP macro in the following program be expanded to on preprocessing? Will the code compile?

```
#include <stdio.h>
#define SWAP( a, b, c ) ( c t ; t = a, a = b, b = t ; )
int main( )
{
    int x = 10, y = 20 ;
    SWAP ( x, y, int ) ;
    printf ( "%d %d\n", x, y ) ;
    return 0 ;
}
```

Answer

```
( int t ; t = a, a = b, b = t ; ) ;
```

This code won't compile since declaration of *t* cannot occur within parentheses.

Question 6.7

How will you modify the SWAP macro in 6.6 above such that it is able to exchange two integers?

Answer

```
#define SWAP( a, b, c ) c t ; t = a, a = b, b = t ;
```

Question 6.8

What is the limitation of the SWAP macro of 6.7 above?

Answer

It cannot swap pointers. For example, the following code will not compile.

```
#include <stdio.h>
#define SWAP( a, b, c ) c t ; t = a, a = b, b = t ;
int main( )
{
    float x = 10, y = 20 ;
    float *p, *q ;
    p = &x ; q = &y ;
    SWAP ( p, q, float* ) ;
    printf ( "%f %f\n", x, y ) ;
    return 0 ;
}
```

Question 6.9

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

```
#include <stdio.h>
#define SWAP(a, b) int t ; t = a ; a = b ; b = t ;
int main( )
{
    int a = 10, b = 12 ;
    SWAP ( a, b ) ;
    printf ( "a = %d b = %d\n", a, b ) ;
    return 0 ;
}
```

A. a = 10 b = 12
B. a = 12 b = 10
C. Error: Declaration is not allowed in a macro
D. Error: Undefined symbol 't'

Answer

B

Question 6.10

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

```
#include <stdio.h>
#define SI(p,n,r) float si ; si = p * n * r / 100 ;
int main( )
{
    float p = 2500, r = 3.5 ;
    int n = 3 ;
    SI ( p, n, r ) ;
    SI ( 1500, 2, 2.5 ) ;
    return 0 ;
}
```

Answer

This code will generate the following error:

- Multiple declarations for si

To remove this error, modify the macro as shown below:

```
#define SI( p, n, r )    p * n * r / 100
```

Question 6.11

Which of the following are correct preprocessor directives in C?

A. #ifdef
B. #if
C. #elif

D. #undef
E. #pragma

Answer

A, B, C, D, E

Question 6.12

What will be the output of the following program?

```
#include <stdio.h>
#define FUN( i, j ) i##j
int main( )
{
    int val1 = 10 ;
    int val12 = 20 ;
    printf ( "%d\n", FUN ( val1, 2 ) ) ;
    return 0 ;
}
```

A. 10
B. 20
C. 1020
D. 12

Answer

B

Question 6.13

Which line in the following program will report an error?

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

```
2.  #define CIRCUM(R) ( 3.14 * R * R ) ;
3.  int main( )
4.  {
5.      float r = 1.0, c ;
6.      c = CIRCUM ( r ) ;
7.      printf ( "%f\n", c ) ;
8.      if ( CIRCUM ( r ) == 6.28 )
9.              printf ( "Gobbledygook\n" ) ;
10.     return 0 ;
11. }
```

Answer

An error will be reported in line number 8, whereas the culprit is really the semicolon in line number 2. On expansion line 8 becomes *if ((3.14 * 1.0 * 1.0) ; == 6.28)*. Hence the error.

Question *6.14*

What is the type of the variable *b* in the following declaration?

```
#define FLOATPTR float *
FLOATPTR a, b ;
```

Answer

float and not a pointer to a *float*, since on expansion the declaration becomes:

```
float *a, b ;
```

Question *6.15*

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


```
#include <stdio.h>
int main( )
{
    int i ;
    #if  A
        printf ( "Enter any number: \n" ) ;
        scanf ( "%d", &i ) ;
    #elif  B
        printf ( "The number is odd\n" ) ;
    return 0 ;
}
```

Answer

Error: 'Unexpected end of file found' because there is no matching *#endif*.

Question *6.16*

What will be the output of the following program?

```
#include <stdio.h>
#define MAN( x, y ) ( ( x ) > ( y ) ) ? ( x ) : ( y ) ;
int main( )
{
    int i = 10, j, k ;
    j = 5 ;
    k = 0 ;
    k = MAN ( ++i, j++ ) ;
    printf ( "%d %d %d\n", i, j, k ) ;
    return 0 ;
}
```

A. 12 6 12
B. 11 5 11
C. 11 5 Garbage

D. 12 6 Garbage

Answer

A

Question 6.17

What will be the output of the following program?

```
#include <stdio.h>
#define FUN( arg )    do \
                      { \
                          if ( arg )\
                              printf ( "Have fun...","\n" ) ;\
                      } while ( i-- )
int main( )
{
    int i = 2 ;
    FUN ( i < 3 ) ;
    return 0 ;
}
```

A. Have fun...
   Have fun...
   Have fun...
B. Have fun...Have fun...Have fun...
C. Error: Cannot use control instructions in a macro
D. No output

Answer

B

Question 6.18

What will be the output of the following program?

```
#include <stdio.h>
#define JOIN( s1, s2 ) printf ( "%s = %s %s = %s\n", #s1, s1, #s2, s2 )
int main( )
{
    char *str1 = "KICIT" ;
    char *str2 = "Quest" ;
    JOIN ( str1, str2 ) ;
    return 0 ;
}
```

Answer

str1 = KICIT str2 = Quest

Question 6.19

What will be the output of the following program?

```
#include <stdio.h>
#define MIN( x, y ) ( x < y ) ? x : y
int main( )
{
    int x = 3, y = 4, z ;
    z = MIN ( x + y / 2, y - 1 ) ;
    if ( z > 0 )
        printf ( "%d\n", z ) ;
    return 0 ;
}
```

A. 3
B. 4

C. 0
D. No output

Answer

A

Question *6.20*

Will the following program compile successfully?

```
#include <stdio.h>
#define X ( 4 + Y )
#define Y ( X + 3 )
int main( )
{
    printf ( "%d\n", 4 * X + 2 ) ;
    return 0 ;
}
```

Answer

No. It will report the following error:

Error: Undefined symbol 'X'

Question *6.21*

Is it necessary that the header files should have a .h extension?

Answer

No. However, traditionally they have been given a .h extension to identify them as something different than the .c program files.

Question *6.22*

What do the header files usually contain?

Answer

Header files contain Preprocessor directives like *#define*, *structure*, *union* and *enum* declarations, *typedef* declarations, global variable declarations and external function declarations. You should not write the actual code (i.e. function bodies) or global variable definition (that is defining or initialising instances) in header files. The *#include* directive should be used to pull in header files, not other '.c' files.

Question *6.23*

Will it result into an error if a header file is included twice? [Yes/No]

Answer

Yes, unless the header file has taken care to ensure that if already included it doesn't get included again. How this can be achieved is shown in *Question 6.24*.

Question *6.24*

How can a header file ensure that it doesn't get included more than once?

Answer

All declarations must be written in the manner shown below. Assume that the name of the header file is 'FUNCS.H'.

```
/* funcs.h */
#ifndef _FUNCS
```

```
#define _FUNCS
/* all declarations would go here */
#endif
```

Now if we include this file twice as shown below, it will get included only once.

```
#include "funcs.h"
#include "funcs.h"
int main( )
{
    /* some code */
    return 0 ;
}
```

Question 6.25

On doing *#include* where are the header files searched?

Answer

If *#include*d using < > the files get searched in the predefined include path. It is possible to change the predefined include path.

If *#include*d with the " " syntax in addition to the predefined include path the files also get searched in the current directory (usually the directory from which you invoked the compiler).

Question 6.26

Would the following *typedef* work?

```
typedef #include I ;
```


Answer

No. Because *typedef* goes to work after preprocessing.

Question 6.27

Will the following code compile correctly?

```
#include <stdio.h>
int main( )
{
    #ifdef NOTE
        /* unterminated comment
        int a ;
        a = 10 ;
    #else
        int a ;
        a = 20 ;
    #endif

    printf ( "%d\n", a ) ;
    return 0 ;
}
```

Answer

No. Even though the *#ifdef* fails in this case (NOTE being undefined) and the *if* block doesn't go for compilation, errors in the *if* block are not permitted.

Question 6.28

What will be the output of the following program?

```
#include <stdio.h>
#define PLANCK 6.626E-34
```

```
int main( )
{
    printf ( "PLANCK\n" ) ;
    return 0 ;
}
```

Answer

PLANCK

Question 6.29

Will the following program print the message infinite number of times? [Yes/No]

```
#include <stdio.h>
#define INFINITELOOP  while ( 1 )
int main( )
{
    INFINITELOOP
    printf ( "Grey haired\n" ) ;
    return 0 ;
}
```

Answer

Yes

Question 6.30

What will be the output of the following program?

```
#include <stdio.h>
#define MAX( a, b )  ( a > b ? a : b )
int main( )
```



```
{
    int x ;
    x = MAX ( 3 + 2, 2 + 7 ) ;
    printf ( "%d\n", x ) ;
    return 0 ;
}
```

Answer

9

Question 6.31

What will be the output of the following program?

```
#include <stdio.h>
#define PRINT( int )  printf ( "%d ", int )
int main( )
{
    int x = 2, y = 3, z = 4 ;
    PRINT ( x ) ;
    PRINT ( y ) ;
    PRINT ( z ) ;
    printf ( "\n" ) ;
    return 0 ;
}
```

Answer

2 3 4

Question 6.32

What will be the output of the following program?

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

```
#define PRINT( int ) printf ( "int = %d ", int )
int main( )
{
    int x = 2, y = 3, z = 4 ;
    PRINT ( x ) ;
    PRINT ( y ) ;
    PRINT ( z ) ;
    printf ( "\n" ) ;
    return 0 ;
}
```

Answer

int = 2 int = 3 int = 4

Question *6.33*

How will you modify the macro of 6.32 such that it outputs:

```
x = 2 y = 3 z = 4
```

Answer

```
#include <stdio.h>
#define PRINT( int ) printf ( #int " = %d ", int )
int main( )
{
    int x = 2, y = 3, z = 4 ;
    PRINT ( x ) ;
    PRINT ( y ) ;
    PRINT ( z ) ;
    printf ( "\n" ) ;
    return 0 ;
}
```


The rule is, if the parameter name is preceded by a # in the macro expansion, the combination (of # and parameter) will be expanded into a quoted string with the parameter replaced by the actual argument. This can be combined with string concatenation to print the output desired in our program. On expansion the macro becomes

```
printf ( "x" " = %d", x ) ;
```

The two strings get concatenated, so the effect is

```
printf ( "x = %d", x ) ;
```

Question *6.34*

Will the following program compile successfully? [Yes/No]

```
#include <stdio.h>
int main( )
{
    printf ( "Tips" "Traps\n" ) ;
    return 0 ;
}
```

Answer

Yes. The output will be TipsTraps. In fact this result has been used in 6.33 above.

Question *6.35*

How will you define the macro DEBUG if the program given below is to produce the following output?

```
DEBUG: x = 4
DEBUG: y = 3.140000
```

```
DEBUG: ch = A
```

```
#include <stdio.h>
int main( )
{
    int x = 4 ;
    float a = 3.14 ;
    char ch = 'A' ;
    DEBUG ( x, %d ) ;
    DEBUG ( a, %f ) ;
    DEBUG ( ch, %c ) ;
    return 0 ;
}
```

Answer

```
#define DEBUG( var, fmt ) printf ( "DEBUG:" #var " = " #fmt "\n", var )
```

Question *6.36*

What will be the output of the following program?

```
#include <stdio.h>
#define str(x) #x
#define Xstr(x) str(x)
#define oper multiply
int main( )
{
    char *opername = Xstr ( oper ) ;
    printf ( "%s\n", opername ) ;
    return 0 ;
}
```

Answer

multiply

Here two operations are being carried out—expansion and stringizing. *Xstr( )* macro expands its argument, and then *str( )* stringizes it.

Question *6.37*

Write the macro PRINT for the following program such that it outputs:

```
x = 4 y = 4 z = 5
a = 1 b = 2 c = 3
```

```
#include <stdio.h>
int main( )
{
    int x = 4, y = 4, z = 5 ;
    int a = 1, b = 2, c = 3 ;
    PRINT ( x, y, z ) ;
    PRINT ( a, b, c ) ;
    printf ( "\n" ) ;
    return 0 ;
}
```

Answer

```
#define PRINT( var1, var2, var3 ) printf ( "\n" #var1 " = %d " #var2 " = %d " \
                                           #var3 " = %d ", var1, var2, var3 )
```

Question *6.38*

Which of the following is the stage at which the statement

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

gets replaced by the contents of the file *stdio.h*?

A. During editing
B. During linking

C. During execution
D. During preprocessing
E. During compilation

Answer

D

Question 6.39

How will you use the macros defined in the following program to test whether a given character is an alphabet or not?

```
#include <stdio.h>
#define LE <=
#define GE >=
#define AND &&
#define OR ||
int main( )
{
    char ch = 'D' ;
    /* write code here to test whether ch contains an alphabet */
    return 0 ;
}
```

Answer

```
if ( ( ch GE 65 AND ch LE 90 ) OR ( ch GE 97 AND ch LE 122))
    printf ( "Alphabet \n" ) ;
else
    printf ( "Not an alphabet \n" ) ;
```


Question 6.40

Which of the following is the correct output for the C code snippet given below?

```
#include <stdio.h>
#define MAX(a, b, c) ( a > b ? a > c ? a : c : b > c ? b : c )
int main( )
{
    int x ;
    x = MAX( 3 + 2, 2 + 7, 3 + 7 ) ;
    printf ( "%d\n", x ) ;
    return 0 ;
}
```

A. 5
B. 9
C. 10
D. 3 + 7

Answer

C

Question 6.41

Which of the following are correctly formed *#define* statements in C?

A. #define CUBE (X) (X*X*X)
B. #define CUBE(x) (x*x*x)
C. #define CUBE (X) (X*X*X) ;
D. #define CUBE(X) (X) * (X) * (X)

Answer

D

Question 6.42

What will be the output of the program given below?

```
#include <stdio.h>
#define PI  3.14

void fun( ) ;
int main( )
{
    printf ( "%f\n", PI ) ;
    #define PI  3.141528
    fun( ) ;
    return 0 ;
}
void fun( )
{
    printf ( "%f\n", PI ) ;
}
```

A. 3.140000
   3.141528
B. 3.140000
   3.140000
C. 3.141528
   3.141528
D. 3.141528
   3.140000

Answer

A

Question 6.43

How would you generate the expanded source code while using the TC, Visual Studio and gcc compilers?

Answer

Carry out the following steps to generate the expanded source code in TC:

- Go to C:\> prompt using Alt F | DOS Shell
- Execute the C Preprocessor using the command:

```
C:\> cpp  PR1.C
```

This command will generate the file PR1.I which contains the expanded source code. You can then open this file in TC to view the expanded source code.

Carry out the following steps to generate the expanded source code in Visual Studio:

- Go to C:\> prompt by invoking the command shell using the cmd command
- Compile the program at C:\> prompt using the command:

```
C:\>CL  /P  PR1.C
```

This command will generate the expanded source code and store it in PR1.I.

To generate the expanded source code in gcc execute the command:

```
$ gcc  -E  PR1.C
```

This command will generate the expanded source code and display it on the screen. If you wish you can redirect the output to a file PR1.I by using the command as shown below.

```
$ gcc -E PR1.C > PR1.I
```