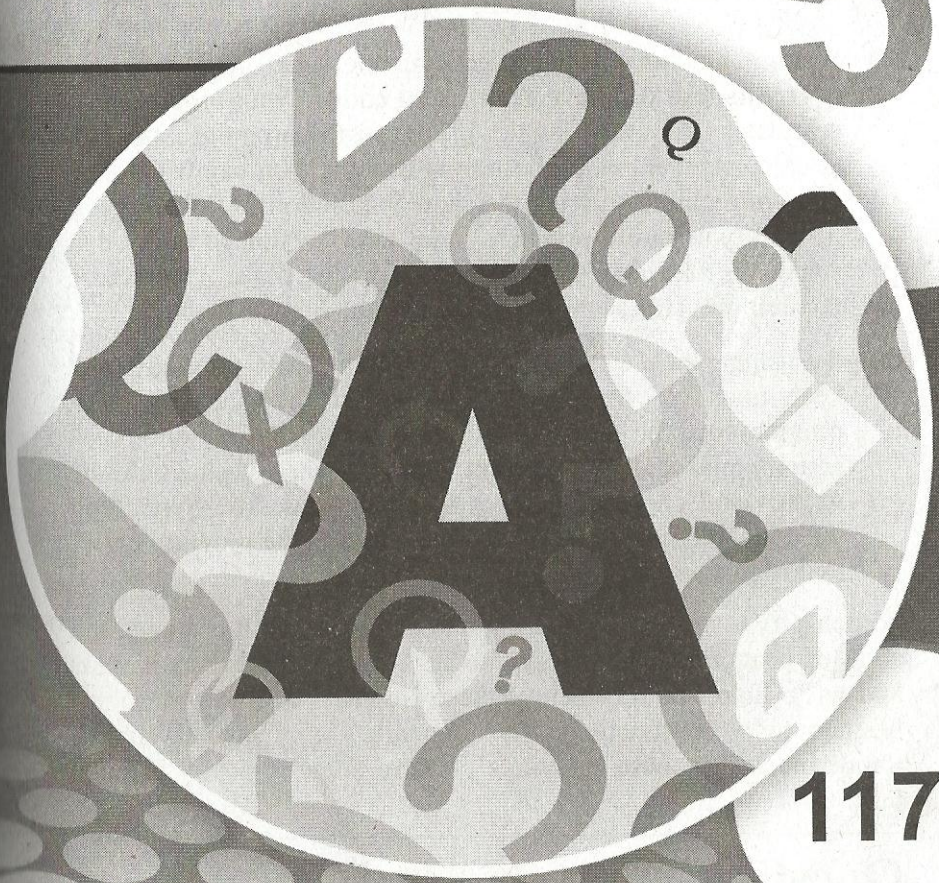
Functions

5



Question 5.1

Why should I use functions at all?

Answer

There are two reasons for using functions:

(a) Writing functions avoids rewriting the same code over and over. Suppose you have a section of code in your program that calculates area of a triangle. If later in the program you want to calculate the area of a different triangle, you won't like it if you are required to write the same instructions all over again. Instead, you would prefer to jump to a 'section of code' that calculates area and then jump back to the place from where you left off. This section of code is nothing but a function.

(b) By using functions it becomes easier to write programs and keep track of what they are doing. If the operation of a program can be divided into separate activities, and each activity placed in a different function, then each could be written and checked more or less independently. Separating the code into modular functions also makes the program easier to design and understand.

So don't try to cram the entire logic in one function. It is a very bad style of programming. Instead, break a program into small units and write functions for each of these isolated subdivisions. Don't hesitate to write functions that are called only once. What is important is that these functions perform some logically isolated task.

Question 5.2

Does Windows use functions?

Answer

Yes. Entire Windows Application Programming Interface (API) is nothing but a collection of C functions. For example, to create a window, there is an API function called *CreateWindow( )*, whereas to display the window on the screen, there is a function called *ShowWindow( )*.

Question 5.3

In what form are the library functions provided?

Answer

Library functions are never provided in source code form. They are always made available in object code form obtained after compilation.

Question 5.4

What will be the output of the following program?

```
#include <stdio.h>
int sumdig ( int ) ;
int main( )
{
    int a, b ;
    a = sumdig ( 123 ) ;
    b = sumdig ( 123 ) ;
    printf ( "%d %d\n", a, b ) ;
    return 0 ;
}
int sumdig ( int n )
{
    int s, d ;
    if ( n != 0 )
    {
```

```
        d = n % 10 ;
        n = n / 10 ;
        s = d + sumdig ( n ) ;
    }
    else
        return ( 0 ) ;
    return ( s ) ;
}
```

Answer

6 6

Question 5.5

What error will the following function give on compilation?

```
f ( int a, int b )
{
    int a ;
    a = 20 ;
    return a ;
}
```

A. Missing parentheses in *return* statement
B. The function should be defined as *int f (int a, int b)*
C. Redeclaration of *a*
D. None of the above

Answer

C

Question 5.6

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

```
#include <stdio.h>
void fun ( int *, int * ) ;
int main( )
{
    int i = 5, j = 2 ;
    fun ( &i, &j ) ;
    printf ( "%d %d\n", i, j ) ;
    return 0 ;
}
void fun ( int *i, int *j )
{
    *i = *i * *i ;
    *j = *j * *j ;
}
```

A. 5 2
B. 10 4
C. 2 5
D. 25 4
E. 4 25

Answer

D

Question 5.7

There is a mistake in the following code. Which statement will you add to remove it?

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

```
int main( )
{
    int a ;
    a = f ( 10, 3.14 ) ;
    printf ( "%d\n", a ) ;
    return 0 ;
}
float f ( int aa, float bb )
{
    return ( ( float ) aa + bb ) ;
}
```

Answer

Add the following function prototype either in *main( )* or above it:

```
float f ( int, float ) ;
```

Question 5.8

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

```
#include <stdio.h>
int main( )
{
    int a = 10 ;
    void f( ) ;
    a = f( ) ;
    printf ( "%d\n", a ) ;
    return 0 ;
}
void f( )
{
    printf ( "\nHi" ) ;
}
```


Answer

In spite of defining the function *f( )* as returning *void*, the program is trying to collect the value returned by *f( )* in the variable *a*.

Question 5.9

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

```
#include <stdio.h>
int main( )
{
    int f ( int ) ;
    int b ;
    b = f ( 20 ) ;
    printf ( "%d\n", b ) ;
    return 0 ;
}
int f ( int a )
{
    a > 20 ? return ( 10 ) : return ( 20 ) ;
}
```

Answer

return statement cannot be used as shown with the conditional operators. Instead the following statement can be used:

```
return ( a > 20 ? 10 : 20 ) ;
```

Question 5.10

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

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

```
{
    int i = 1 ;
    if ( !i )
        printf ( "Recursive calls are painful\n" ) ;
    else
    {
        i = 0 ;
        printf ( "Recursive calls are challenging\n" ) ;
        main( ) ;
    }
    return 0 ;
}
```

A. Recursive calls are challenging
   Recursive calls are painful
B. Recursive calls are painful
   Recursive calls are challenging
C. Recursive calls are real pain!
D. Recursive calls are challenging
E. The code prints
   Recursive calls are challenging
   Recursive calls are challenging (infinitely...)

Answer

E

Question 5.11

State True or False:

A. In C all functions except *main( )* can be called recursively.

B. A function CANNOT be defined inside another function.

C. Functions can be called either by value or by reference.

D. Functions cannot return more than one value at a time.

E. Names of functions in two different files linked together must be unique.

F. Every function must return a value.

G. Maximum number of arguments that a function can take is 12.

H. A function may have any number of return statements each returning different values.

I. If return type for a function is not specified, it defaults to int.

J. Functions cannot return a floating point number.

K. While defining a function it is necessary to collect the values passed to it.

L. It is not compulsory to collect the value returned from a function.

M. Formal arguments used in a function can be variables, constants or expressions.

N. Actual arguments used in a function can be variables, constants or expressions.

O. Redeclaration of a function is not an error.

P. Redefinition of a function is an error.

Answer

A. False. Any function including *main( )* can be called recursively.
B. True
C. True
D. True
E. True
F. False
G. False
H. True
I. True
J. False
K. True
L. True
M. False
N. True
O. True
P. True

Question *5.12*

Will the following functions work? [Yes/No]

```
int f1 ( int a, int b )
{
    return ( f2 ( 20 ) ) ;
}
int f2 ( int a )
{
    return ( a * a ) ;
}
```


Answer

Yes. *f1( )* returns the value returned to it by *f2( )*.

Question *5.13*

What are the following two notations of defining functions commonly known as:

```
inf f ( int a, float b )
{
    /* some code */
}

int f ( a, b )
int a ; float b ;
{
    /* some code */
}
```

Answer

The first one is known as ANSI notation and the second is known as Kernighan and Ritchie, or simply, K & R notation. Many modern compilers support only ANSI notation.

Question *5.14*

In a function two *return* statements should never occur. [True/False]

Answer

False. For example, the following function works correctly:

```
int f ( int a )
```

```
{
    if ( a > 20 )
        return 10 ;
    else
        return 20 ;
}
```

Question 5.15

In a function two *return* statements should never occur successively. [True/False]

Answer

True. For example, in the following function the second *return* statement is redundant.

```
int f ( int a )
{
    return 10 ;
    return 20 ;
}
```

Question 5.16

Which of the following statements are correct about the following program?

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

A. It prints "main" infinite number of times.

B. It runs an infinite loop without printing anything.
C. Compiler reports an error since *main( )* cannot be called recursively.
D. Address of *main( )* would get printed infinite number of times.
E. Address of *main( )* would get printed once.

Answer

E

Question 5.17

Usually recursion works slower than loops. [True/False]

Answer

True

Question 5.18

Is it true that too many recursive calls may result into stack overflow? [Yes/No]

Answer

Yes

Question 5.19

How many times the following program will print "Jamboree"?

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

```
}
```

A. Infinite number of times
B. 32767 times
C. 65535 times
D. Till the stack doesn't overflow

Answer

D

Question *5.20*

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

```
#include <stdio.h>
void fun ( char * ) ;
int main( )
{
    char a[100] ;
    a[0] = 'A', a[1] = 'B' ;
    a[2] = 'C', a[3] = 'D' ;
    fun ( &a[0] ) ;
    return 0 ;
}
void fun ( char *a )
{
    a++ ;
    printf ( "%c", *a ) ;
    a++ ;
    printf ( "%c\n", *a ) ;
}
```

A. AB
B. BC

C. CD
D. No output

Answer

B

Question *5.21*

What will be the output of the following program?

```
#include <stdio.h>
int main( )
{
    int fun( ) ;
    int i ;
    i = fun( ) ;
    printf ( " %d\n ", i ) ;
    return 0 ;
}
int fun( )
{
    _AX = 1990 ;
    return 0 ;
}
```

Answer

1990. The return value of a function is taken from the accumulator. Here *_AX* is the pseudo global variable, denoting the accumulator. This code works in TC/TC++. However, it is non-portable code and may not work with other compilers.

Question 5.22

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

```
#include <stdio.h>
int main( )
{
    int fun ( int ) ;
    int i = fun ( 10 ) ;
    printf ( "%d\n", --i ) ;
    return 0 ;
}
int fun ( int i )
{
    return ( i++ ) ;
}
```

A. 9
B. 10
C. 11
D. 8

Answer

A

Question 5.23

How many sets of variables are created if *fun( )* is called from *main( )* by passing a value 1 to it?

```
void fun( int i )
{
    int j ;
    j = i ;
    j++ ;
    i++ ;
    if ( i <= 3 )
        fun ( i ) ;
}
```


Answer

3 sets of *i* and *j* are created, 1 set per call.

Question 5.24

When would the variable *k* die in the following function?

```
int fun( )
{
    int i = 1, j ;
    j = i * 2 ;
    if ( j <= 5 )
    {
        int k = 2 ;
        k++ ;
        printf ( "%d\n", k ) ;
    }
    printf ( "%d\n", i ) ;
    printf ( "%d\n", j ) ;
    return 0 ;
}
```

Answer

Variable *k* would die as soon as control goes outside the *if* block.

Question 5.25

How would you prove that in the following program 3 sets of *i* and *j* are created?

```
#include <stdio.h>
void fun ( int ) ;
int main( )
{
    fun ( 1 ) ;
    return 0 ;
}

void fun ( int i )
{
    int j ;
    j = i ;
    j++ ;
    i++ ;
    if ( i <= 3 )
        fun ( i ) ;
}
```

Answer

Add a statement

```
printf ( "%u %u\n", &i, &j ) ;
```

after the statement *i++ ;* in function *fun( )*.

During each call a different set of addresses would get printed. This proves that there are different sets of variables in existence.

Question 5.26

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

```
#include <stdio.h>
int fun ( int, int ) ;
typedef int (*pf ) ( int, int ) ;
int proc ( pf, int, int ) ;

int  main( )
{
    printf ( "%d\n", proc ( fun, 6, 6 ) ) ;
    return 0 ;
}

int fun ( int a, int b )
{
    return ( a == b ) ;
}

int proc ( pf p, int a, int b )
{
    return ( (*p) ( a, b ) ) ;
}
```

A. 6
B. 1
C. 0
D. -1

Answer

B

Question 5.27

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

```
#include <stdio.h>
void fun ( int ) ;
int main( )
{
    int a ;
    a = 3 ;
    fun ( a ) ;
    printf ( "\n" ) ;
    return 0 ;
}
void fun ( int n )
{
    if ( n > 0 )
    {
        fun ( --n ) ;
        printf ( "%d ", n ) ;
        fun ( --n ) ;
    }
}
```

A. 0 2 1 0
B. 1 1 2 0
C. 0 1 0 2
D. 0 1 2 0

Answer

D

Question 5.28

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

```
#include <stdio.h>
int func1 ( int ) ;
int main( )
{
    int k = 35 ;
    k = func1 ( k = func1 ( k = func1 ( k) ) ) ;
    printf ( "k = %d\n", k ) ;
    return 0;
}
int func1 ( int k )
{
    k++ ;
    return ( k ) ;
}
```

A. k = 35
B. k = 36
C. k = 37
D. k = 38
E. k = 39

Answer

D

Question 5.29

If an *int* is 4 bytes wide then which of the following is the correct output for the program given below?

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

```
void fun ( char ** ) ;
int main( )
{
    char *argv[ ] = { "ab", "cd", "ef", "gh" } ;
    fun ( argv ) ;
    return 0 ;
}
void fun ( char **p )
{
    char *t ;
    t = ( p += sizeof ( int ) ) [ -1 ] ;
    printf ( "%s \n", t ) ;
}
```

A. ab
B. cd
C. ef
D. gh

Answer

D

Question *5.30*

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

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



```
int fun ( int i )
{
    i++ ;
    return ( i ) ;
}
```

A. 5
B. 4
C. Error
D. Garbage value

Answer

A

Question *5.31*

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

```
#include <stdio.h>
float fun ( float ) ;
int main( )
{
    float k = 0.5 ;
    fun ( k = fun ( fun ( k ) ) ) ;
    printf ( "%f \n", k ) ;
    return 0 ;
}
float fun ( float i )
{
    return i * i ;
}
```

A. 0.062500
B. 6.25

C. Garbage value
D. Error

Answer

A

Question 5.32

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

```
#include <stdio.h>
int i ;
int fun1 ( int ) ;
int fun2 ( int ) ;
int main( )
{
    extern int j ;
    int i = 3 ;
    fun1 ( i ) ;
    printf ( " %d ", i ) ;
    fun2 ( i ) ;
    printf ( " %d \n", i ) ;
    return 0 ;
}
int fun1 ( int j )
{
    printf ( " %d ", ++j ) ;
    return 0 ;
}
int fun2 ( int i )
{
    printf ( " %d ", ++i ) ;
    return 0 ;
}
int j = 1 ;
```


A. 3 4 4 3
B. 4 3 4 3
C. 3 3 4 4
D. 3 4 3 4

Answer

B

Question 5.33

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

```
#include <stdio.h>
int fun ( int ( * )( ) ) ;
int main( )
{
    fun ( main ) ;
    printf ( "Hi \n" ) ;
    return 0 ;
}
int fun ( int ( *p )( ) )
{
    printf ( "Hello " ) ;
    return 0 ;
}
```

A. An infinite loop
B. Hi
C. Hello Hi
D. Error

Answer

C

Question 5.34

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

```
#include <stdio.h>
int fun( ) ;
int i ;
int main( )
{
    while ( i )
    {
        fun( ) ;
        main( ) ;
    }
    printf ( "Hello\n" ) ;
    return 0 ;
}
int fun( )
{
    printf ( "Hi" ) ;
    return 0 ;
}
```

A. Hello
B. Hi Hello
C. No output
D. Infinite loop

Answer

A

Question 5.35

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

```
#include <stdio.h>
#include <stdlib.h>
int main( )
{
    int i = 0 ;
    i++ ;
    if ( i <= 5 )
    {
        printf ( "adds wings to your thoughts\n" ) ;
        exit ( 1 ) ;
        main( ) ;
    }
    return 0 ;
}
```

A. The code prints 'adds wings to your thoughts' five times.
B. Function *main( )* cannot call itself.
C. The code generates infinite loop.
D. The code prints 'adds wings to your thoughts'

Answer

D

Question 5.36

Write a recursive function *count( )* that prints numbers from 10 to 1.

Answer

```
int count ( int no )
{
    if ( no < 1 )
        return 0 ;
    else
        printf ( "%d", no ) ;
    count (--no ) ;
}
```

Question 5.37

The keyword used to transfer control from a function back to the calling function is:

A. switch
B. goto
C. go back
D. return

Answer

D

Question 5.38

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

```
#include <stdio.h>
int addmult ( int, int ) ;
int main( )
{
    int i = 3, j = 4, k, l ;
    k = addmult ( i, j ) ;
```



```
    l = addmult ( i, j ) ;
    printf ( "%d %d\n", k, l ) ;
    return 0 ;
}
int addmult ( int ii, int jj )
{
    int kk, ll ;
    kk = ii + jj ;
    ll = ii * jj ;
    return ( kk, ll ) ;
}
```

A. 12 12
B. 7 7
C. 7 12
D. 12 7
E. 3 4

Answer

A

The comma-separated expression *(kk, ll)* evaluates to *ll*. Value of *ll* is 12. Hence in both the calls to *addmult( )* 12 is returned.

Question 5.39

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

```
#include <stdio.h>
int check ( int ) ;
int main( )
{
    int i = 45, c ;
    c = 2 * check ( i ) + check ( i ) ;
    printf ( "%d\n", c ) ;
```

```
    return 0 ;
}
int check ( int ch )
{
    if ( ch >= 45 )
        return ( 100 ) ;
    else
        return ( 10 ) ;
}
```

A. 300
B. 100
C. Error: Call being used in an arithmetic expression
D. 30

Answer

A

Question *5.40*

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

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



```
    q = &j ;
    i >= 45 ? return ( *p ) : return ( *q ) ;
}
```

A. Program outputs a value 10.
B. Program outputs a value 20.
C. Program outputs a value 1.
D. Compiler reports an error since *return* cannot be used with ? :.

Answer

D

Question *5.41*

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

```
#include <stdio.h>
int reverse ( int ) ;
int main( )
{
    int no = 5 ;
    reverse ( no ) ;
    printf ( "\n" ) ;
    return 0 ;
}
int reverse ( int no )
{
    if ( no == 0 )
        return 0 ;
    else
        printf ( "%d ", no ) ;
    reverse ( no-- ) ;
}
```

A. Program outputs values 5 4 3 2 1.

B. Program outputs values 1 2 3 4 5.
C. Program outputs values 5 4 3 2 1 0.
D. Program runs in an infinite loop.
E. Program prints 5 till stack doesn't overflow.

Answer

E

Question 5.42

Which of the following statements are correct about the function given below?

```
long fun ( int num )
{
    int i ;
    long f = 1 ;
    for ( i = 1 ; i <= num ; i ++ )
        f = f * i ;
    return ( f ) ;
}
```

A. The function calculates the value of *1* raised to power *num*.
B. The function calculates the square root of an integer.
C. The function calculates the factorial value of an integer.
D. None of the above.

Answer

C

Question 5.43

In the following program is it necessary to mention prototype of function *fun( )*?


```
#include <stdio.h>
void fun( )
{
    printf ( "Hello\n" ) ;
}

int main( )
{
    fun( ) ;
    return 0 ;
}
```

Answer

In the above program *fun( )* is defined before it is called from *main( )*. Hence it is not necessary to mention its prototype. Had function *fun( )* been defined after the call to it then it would have been necessary to mentions its prototype.

Question 5.44

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

```
#include <stdio.h>
int main( )
{
    void fun1 ( int ) ;
    void fun2 ( int ) ;
    int fun3 ( int ) ;
    int num = 5 ;
    fun1 ( num ) ;
    fun2 ( num ) ;
    return 0 ;
}
```

```
void fun1 ( int no )
{
    no++ ;
    fun3 ( no ) ;
}

void fun2 ( int no )
{
    no-- ;
    fun3 ( no ) ;
}

void fun3 ( int n )
{
    printf ( "%d ", n ) ;
}
```

A. The program would produce an output 6 4.
B. The program cannot compile successfully unless prototype of *fun3( )* is shifted outside *main( )*.
C. The program cannot compile successfully unless prototype of *fun1( )*, *fun2()* and *fun3( )* are shifted outside *main( )*.
D. Function Prototypes cannot be declared inside *main( )*.

Answer

B