Declarations and Initializations

General Questions

1. Which of the following statements should be used to obtain a remainder after dividing 3.14 by 2.1?
   A. rem = 3.14 % 2.1;
   B. rem = modf(3.14, 2.1);
   C. rem = fmod(3.14, 2.1);
   D. Remainder cannot be obtained in floating point division.

   Answer: Option C

   Explanation:

   \textit{fmod}(x, y) - Calculates x modulo y, the remainder of x/y. This function is the same as the modulus operator. But \textit{fmod()} performs floating point divisions.

   Example:

   ```c
   #include <stdio.h>
   #include <math.h>
   
   int main ()
   { 
       printf ("fmod of 3.14/2.1 is %lf\n", fmod (3.14, 2.1));
       return 0;
   }
   ```

2. What are the types of linkages?
   A. Internal and External
   B. External, Internal and None
   C. External and None
   D. Internal

   Answer: Option B

   Explanation:

   External Linkage-> means global, non-static variables and functions.
   Internal Linkage-> means static variables and functions with file scope.
   None Linkage-> means Local variables.

3. Which of the following special symbol allowed in a variable name?
   A. * (asterisk)
   B. | (pipeline)
   C. - (hyphen)
   D. _ (underscore)

   Answer: Option D
Explanation:

Variable names in C are made up of letters (upper and lower case) and digits. The underscore character ("_") is also permitted. Names must not begin with a digit.

Examples of valid (but not very descriptive) C variable names:
=> foo
=> Bar
=> BAZ
=> foo_bar
=> _foo42
=> _
=> QuUx

4. Is there any difference between following declarations?

1 : extern int fun();
2 : int fun();

A. Both are identical
B. No difference, except extern int fun(); is probably in another file
C. int fun(); is overridden with extern int fun();
D. None of these

Answer & Explanation

Answer: Option B

Explanation:

extern int fun(); declaration in C is to indicate the existence of a global function and it is defined externally to the current module or in another file.

int fun(); declaration in C is to indicate the existence of a function inside the current module or in the same file.

5. How would you round off a value from 1.66 to 2.0?

A. ceil(1.66)  B. floor(1.66)
C. roundup(1.66)  D. roundto(1.66)

Answer & Explanation

Answer: Option A

Explanation:

/* Example for ceil() and floor() functions: */

#include<stdio.h>
#include<math.h>

int main()
{
    printf("\n Result : %f " , ceil(1.44) );
    printf("\n Result : %f " , ceil(1.66) );
}
printf("\n Result : %f", floor(1.44));
printf("\n Result : %f", floor(1.66));
return 0;
} // Output:
// Result : 2.000000
// Result : 2.000000
// Result : 1.000000
// Result : 1.000000

6. By default a real number is treated as a

A. float
B. double
C. long double
D. far double

Answer & Explanation

Answer: Option B

Explanation:

In computing, 'real number' often refers to non-complex floating-point numbers. It includes both rational numbers, such as 42 and 3/4, and irrational numbers such as pi = 3.14159265...

When the accuracy of the floating point number is insufficient, we can use the double to define the number. The double is same as float but with longer precision and takes double space (8 bytes) than float.

To extend the precision further we can use long double which occupies 10 bytes of memory space.

7. Which of the following is not user defined data type?

1: struct book {
    char name[10];
    float price;
    int pages;
};
2: long int l = 2.35;
3: enum day {Sun, Mon, Tue, Wed};

A. 1
B. 2
C. 3
D. Both 1 and 2

Answer & Explanation

Answer: Option B

Explanation:

C data types classification are

1. Primary data types
   1. int
   2. char
3. float
4. double
5. void

2. Secondary data types (or) User-defined data type
   1. Array
   2. Pointer
   3. Structure
   4. Union
   5. Enum

So, clearly \( \text{long int } l = 2.35 \); is not User-defined data type.
(i.e. \( \text{long int } l = 2.35 \); is the answer.)

8. Is the following statement a declaration or definition?
   \( \text{extern int } i; \)

   **Answer & Explanation**
   **Answer:** Option A
   **Explanation:**
   Declaring is the way a programmer tells the compiler to expect a particular type, be it a variable, class/struct/union type, a function type (prototype) or a particular object instance. (ie. \( \text{extern int } i \))
   Declaration never reserves any space for the variable or instance in the program's memory; it simply a "hint" to the compiler that a use of the variable or instance is expected in the program. This hinting is technically called "forward reference".

9. Identify which of the following are declarations
   1 : \( \text{extern int } x; \)
   2 : \( \text{float square ( float x ) } \{ \ldots \} \)
   3 : \( \text{double pow(double, double);} \)

   **Answer & Explanation**
   **Answer:** Option C
   **Explanation:**
   extern int x; - is an external variable declaration.
   double pow(double, double); - is a function prototype declaration.
   Therefore, 1 and 3 are declarations. 2 is definition.
10. In the following program where is the variable a getting defined and where it is getting declared?

```c
#include<stdio.h>
int main()
{
    extern int a;
    printf("%d\n", a);
    return 0;
}
int a = 20;
```

A. `extern int a` is declaration, `int a = 20` is the definition
B. `int a = 20` is declaration, `extern int a` is the definition
C. `int a = 20` is definition, `a` is not defined
D. `a` is declared, `a` is not defined

Answer: Option A

Explanation:
- During declaration we tell the datatype of the Variable.
- During definition the value is initialized.

11. When we mention the prototype of a function?
A. Defining  
B. Declaring
C. Prototyping  
D. Calling

Answer: Option B

Explanation:
A function prototype in C or C++ is a declaration of a function that omits the function body but does specify the function's name, argument types and return type.
While a function definition specifies what a function does, a function prototype can be thought of as specifying its interface.

**Find Output of Program**

1. What is the output of the program given below?

```c
#include<stdio.h>
int main()
{
enum status { pass, fail, atkt};
enum status stud1, stud2, stud3;
stud1 = pass;
stud2 = atkt;
stud3 = fail;
printf("%d, %d, %d\n", stud1, stud2, stud3);
return 0;
}  
```
A. 0, 1, 2
C. 0, 2, 1
Answer & Explanation

Answer: Option C

Explanation:

enum takes the format like {0,1,2..} so pass=0, fail=1, atkt=2

stud1 = pass (value is 0)
stud2 = atkt (value is 2)
stud3 = fail (value is 1)

Hence it prints 0, 2, 1

2. What will be the output of the program in 16 bit platform (Turbo C under DOS)?

#include<stdio.h>
int main()
{
    extern int i;
    i = 20;
    printf("%d\n", sizeof(i));
    return 0;
}
A. 2
B. 4
C. vary from compiler
D. Linker Error: Undefined symbol 'i'
Answer & Explanation

Answer: Option D

Explanation:

Linker Error: Undefined symbol 'i'
The statement extern int i specifies to the compiler that the memory for 'i' is allocated in some other program and that address will be given to the current program at the time of linking. But linker finds that no other variable of name 'i' is available in any other program with memory space allocated for it. Hence a linker error has occurred.

3. What is the output of the program?

#include<stdio.h>
int main()
{
    extern int a;
    printf("%d\n", a);
    return 0;
}
4. What is the output of the program in Turbo C (in DOS 16-bit OS)?

```
#include<stdio.h>
int main()
{
    char *s1;
    char far *s2;
    char huge *s3;
    printf("%d, %d, %d
", sizeof(s1), sizeof(s2), sizeof(s3));
    return 0;
}
```

**Answer: Option C**

**Explanation:**

Any pointer size is 2 bytes. (only 16-bit offset)

So, char *s1 = 2 bytes.

So, char far *s2 = 4 bytes.

So, char huge *s3 = 4 bytes.

A far, huge pointer has two parts: a 16-bit segment value and a 16-bit offset value.

Since C is a compiler dependent language, it may give different output in other platforms. The above program works fine in Windows (TurboC), but error in Linux (GCC Compiler).

5. What is the output of the program

```
#include<stdio.h>
int main()
{
    struct emp
    {
        char name[20];
        int age;
```
float sal;
};
struct emp e = {"Tiger"};
printf("%d, %f
", e.age, e.sal);
return 0;
}

A. 0, 0.000000
B. Garbage values
C. Error
D. None of above

Answer: Option A

Explanation:

When an automatic structure is partially initialized remaining elements are initialized to 0(zero).

6. What will be the output of the program?

#include<stdio.h>
int X=40;
int main()
{
    int X=20;
    printf("%d
", X);
    return 0;
}

A. 20
B. 40
C. Error
D. No Output

Answer: Option A

Explanation:

Whenever there is conflict between a local variable and global variable, the local variable gets priority.

7. What is the output of the program

#include<stdio.h>
int main()
{
    int x = 10, y = 20, z = 5, i;
    i = x < y < z;
    printf("%d
", i);
    return 0;
}

A. 0
B. 1
C. Error
D. None of these

Answer: Option B
Explanation:

Since \( x < y \) turns to be TRUE it is replaced by 1. Then \( 1 < z \) is compared and to be TRUE. The 1 is assigned to \( i \).

8. What is the output of the program

```c
#include<stdio.h>
int main()
{
    extern int fun(float);
    int a;
    a = fun(3.14);
    printf("%d\n", a);
    return 0;
}
int fun(int aa)
{
    return (int)++aa;
}   
A.3   B.3.14
C.0   D.4
E. Compile Error
Answer & Explanation

Answer: Option E

Explanation:

2 Errors
1. Type mismatch in redeclaration of fun
2. Type mismatch in parameter aa

9. What is the output of the program

```c
#include<stdio.h>
int main()
{
    int a[5] = {2, 3};
    printf("%d, %d, %d\n", a[2], a[3], a[4]);
    return 0;
}   
A. Garbage Values   B. 2, 3, 3
C. 3, 2, 2  D. 0, 0, 0
Answer & Explanation

Answer: Option D

Explanation:

When an automatic array is partially initialized, the remaining elements are initialized to 0.
10. What is the output of the program?

```c
#include<stdio.h>
int main()
{
    union a
    {
        int i;
        char ch[2];
    };
    union a u;
    u.ch[0] = 3;
    u.ch[1] = 2;
    printf("%d, %d, %d\n", u.ch[0], u.ch[1], u.i);
    return 0;
}
A. 3, 2, 515  
B. 515, 2, 3  
C. 3, 2, 5  
D. None of these

Answer & Explanation

Answer: Option A

Explanation:

printf("%d, %d, %d\n", u.ch[0], u.ch[1], u.i); It prints the value of u.ch[0] = 3, u.ch[1] = 2 and it prints the value of u.i means the value of entire union size.

```

So, 512 + 2 + 1 = 515
i = 515

So the output is 3, 2, 515.

11. In the following program how long will the for loop get executed?

```c
#include<stdio.h>
int main()
{
    int i=5;
    for(;scanf("%s", &i); printf("%d\n", i));
    return 0;
}
A. The for loop would not get executed at all  
B. The for loop would get executed only once

Saikat Banerjee
C. The for loop would get executed 5 times
D. The for loop would get executed infinite times

Answer & Explanation

Answer: Option D

Explanation:

During the for loop execution scanf() ask input and then printf() prints that given input. This process will be continued repeatedly because, scanf() returns the number of input given, the condition is always true(user gives a input means it reurns '1').

Hence this for loop would get executed infinite times.

12. What will be the output of the program?

```c
#include<stdio.h>
int main()
{
    int X=40;
    {
        int X=20;
        printf("%d ", X);
    }
    printf("%d
", X);
    return 0;
}
```

A. 40 40
B. 20 40
C. 20

Answer & Explanation

Answer: Option B

Explanation:

In case of a conflict between a local variable and global variable, the local variable gets priority.

**Point Out Errors**

1. Point out the error in the following program.

```c
#include<stdio.h>
int main()
{
    display();
    return 0;
}

void display()
{
    printf("Saikat12.com");
}
```

A. No error
B. display() doesn't get invoked

Saikat Banerjee
C. `display()` is called before it is defined
D. None of these

Answer & Explanation

**Answer:** Option C

**Explanation:**

In this program the compiler will not know that the function `display()` exists. So, the compiler will generate "Type mismatch in redeclaration of function `display()`".

To overcome this error, we need to add function prototype of function `display()`.

Another way to overcome this error is to define the function `display()` before the `int main();` function.

```c
#include<stdio.h>
void display(); /* function prototype */
int main()
{
    display();
    return 0;
}
void display()
{
    printf("Saikat12.com");
}
```

**Output:** Saikat12.com

Note: This problem will not occur in modern compilers (this problem occurs in TurboC but not in GCC).

2. Point out the error in the following program.

```c
#include<stdio.h>
int main()
{
    void v = 0;
    printf("%d", v);
    return 0;
}
```

A. Error: Declaration syntax error 'v' (or) Size of v is unknown or zero.
B. Program terminates abnormally.
C. No error.
D. None of these.

Answer & Explanation

**Answer:** Option A

3. Point out the error in the following program.

```c
#include<stdio.h>
```
struct emp
{
    char name[20];
    int age;
};
int main()
{
    emp int xx;
    int a;
    printf("%d\n", &a);
    return 0;
}

A. Error: in printf
B. Error: in emp int xx;
C. No error.
D. None of these.

Answer: Option B

Explanation:
There is an error in the line emp int xx;
To overcome this error, remove the int and add the struct at the begining of emp int xx;

#include<stdio.h>
struct emp
{
    char name[20];
    int age;
};
int main()
{
    struct emp xx;
    int a;
    printf("%d\n", &a);
    return 0;
}

4. Which of the following is correct about err used in the declaration given below?

typedef enum error { warning, test, exception } err;
A. It is a typedef for enum error.
B. It is a variable of type enum error.
C. The statement is erroneous.
D. It is a structure.

Answer: Option A

Explanation:
A typedef gives a new name to an existing data type.
So err is a new name for enum error.

5. Point out the error in the following program.

Saikat Banerjee
#include<stdio.h>
int main()
{
    int (*p)() = fun;
    (*p)();
    return 0;
}
int fun()
{
    printf("Saikat12.com\n");
    return 0;
}

A. Error: in int(*p)() = fun;
B. Error: fun() prototype not defined
C. No error
D. None of these

Answer: Option B

Explanation:
The compiler will not know that the function int fun() exists. So we have to define the function prototype of int fun();
To overcome this error, see the below program

#include<stdio.h>
int fun(); /* function prototype */
int main()
{
    int (*p)() = fun;
    (*p)();
    return 0;
}
int fun()
{
    printf("Saikat12.com\n");
    return 0;
}

Point Out Correct Statements
1. Which of the declaration is correct?
   A. int length;  
   B. char int;  
   C. int long;  
   D. float double;

Answer: Option A

Explanation:
int length; denotes that variable length is int(integer) data type.
char int; here int is a keyword cannot be used a variable name.
int long; here long is a keyword cannot be used a variable name.

float double; here double is a keyword cannot be used a variable name.

So, the answer is int length;(Option A).

2. Which of the following operations are INCORRECT?
   
   A. int i = 35; i = i%5;  
   B. short int j = 255; j = j;  
   C. long int k = 365L; k = k;  
   D. float a = 3.14; a = a%3;

   **Answer & Explanation**

   **Answer:** Option D

   **Explanation:**

   float a = 3.14; a = a%3; gives "Illegal use of floating point" error.

   The modulus (%) operator can only be used on integer types. We have to use fmod() function in math.h for float values.

3. Which of the following correctly represents a long double constant?

   A. 6.68  
   B. 6.68L  
   C. 6.68f  
   D. 6.68LF

   **Answer & Explanation**

   **Answer:** Option B

   **Explanation:**

   6.68 is double.  
   6.68L is long double constant.  
   6.68f is float constant.  
   6.68LF is not allowed in c.

4. Which of the structure is incorrcet?

   ```c
   struct aa  
   {  
   1 : int a;  
   float b;  
   };  
   struct aa  
   {  
   int a;  
   float b;  
   struct aa var;  
   };  
   struct aa  
   2 :{  
   int a;  
   };  
   ```
float b;
struct aa *var;

A. 1
C. 3

B. 2
D. 1, 2, 3

Answer & Explanation

Answer: Option B

Explanation:

Option B gives "Undefined structure in 'aa"' error.

5. Which of the structure is correct?

<table>
<thead>
<tr>
<th></th>
<th>struct book</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>char name[10];</td>
</tr>
<tr>
<td></td>
<td>float price;</td>
</tr>
<tr>
<td></td>
<td>int pages;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td>2</td>
<td>struct aa</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>char name[10];</td>
</tr>
<tr>
<td></td>
<td>float price;</td>
</tr>
<tr>
<td></td>
<td>int pages;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td>3</td>
<td>struct aa</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>char name[10];</td>
</tr>
<tr>
<td></td>
<td>float price;</td>
</tr>
<tr>
<td></td>
<td>int pages;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

A. 1
C. 3

B. 2
D. All of above

Answer & Explanation

Answer: Option A

Explanation:

In 2 and 3 semicolon are missing in structure element.

6. typedef long a;
   extern int a c;
   typedef long a;
   extern a int c;
   typedef long a;
   extern a c;

A. 1 correct
C. 3 correct
B. 2 correct
D. 1, 2, 3 are correct

Answer & Explanation
Answer: Option C

Explanation:

typedef long a;
extern int a c; while compiling this statement becomes extern int long c; This will result in to "Declaration syntax error".

typedef long a;
extern a int c; while compiling this statement becomes extern long int c; This will result in to "Too many types in declaration error".

typedef long a;
extern a c; while compiling this statement becomes extern long c; This is a valid c declaration statement. It says variable c is long data type and defined in some other file or module.

So, Option C is the correct answer.

**True / False Questions**

1. A long double can be used if range of a double is not enough to accommodate a real number.
   A. True  B. False
   **Answer & Explanation**
   **Answer: Option A**
   **Explanation:**
   True, we can use long double; if double range is not enough.
   
   double = 8 bytes.
   long double = 10 bytes.

2. A float is 4 bytes wide, whereas a double is 8 bytes wide.
   A. True  B. False
   **Answer & Explanation**
   **Answer: Option A**
   **Explanation:**
   True,
   float = 4 bytes.
   double = 8 bytes.

3. If the definition of the external variable occurs in the source file before its use in a particular function, then there is no need for an extern declaration in the function.
   A. True  B. False
   **Answer & Explanation**
Answer: Option A

Explanation:

True, When a function is declared inside the source file, that function(local function) get a priority than the `extern` function. So there is no need to declare a function as `extern` inside the same source file.

4. Size of short integer and long integer can be verified using the `sizeof()` operator.
   A. True
   B. False

Answer & Explanation

Answer: Option A

Explanation:

True, we can find the size of short integer and long integer using the `sizeof()` operator.

Example:

```c
#include<stdio.h>
int main()
{
    short int i = 10;
    long int j = 10;
    printf("short int is %d bytes.,\nlong int is %d bytes.\n",
           sizeof(i),sizeof(j));
    return 0;
}
```

Output:
short int is 2 bytes.
long int is 4 bytes.

5. Range of double is -1.7e-38 to 1.7e+38 (in 16 bit platform - Turbo C under DOS)
   A. True
   B. False

Answer & Explanation

Answer: Option B

Explanation:

False, The range of double is -1.7e-308 to 1.7e+308.

6. Size of short integer and long integer would vary from one platform to another.
   A. True
   B. False

Answer & Explanation

Answer: Option A
Explanation:

True, Depending on the operating system/compiler/system architecture you are working on, the range of data types can vary.

7. Range of float id -2.25e-308 to 2.25e+308
   A. True
   B. False

   Answer: Option B

   Explanation:

   False, The range of float is -3.4e-38 to 3.4e+38.

Yes / No Questions

1. Is there any difference in the following declarations?
   int myfun(int arr[]);
   int myfun(arr[20]);
   A. Yes
   B. No

   Answer: Option A

   Explanation:

   Yes, we have to specify the data type of the parameter when declaring a function.

2. Suppose a program is divided into three files f1, f2 and f3, and a variable is defined in the file f1 but used in files f2 and f3. In such a case would we need the extern declaration for the variables in the files f2 and f3?
   A. Yes
   B. No

   Answer: Option A

3. Global variable are available to all functions. Does there exist a mechanism by way of which it available to some and not to others.
   A. Yes
   B. No

   Answer: Option B

   Explanation:

   The only way this can be achieved is to define the variable locally in main() instead of defining it
globally and then passing it to the functions which need it.

4. Is it true that a global variable may have several declarations, but only one definition?  
   A. Yes  
   B. No  
   **Answer & Explanation**  
   **Answer: Option A**  
   **Explanation:**  
   Yes, in all the global variable declarations, you need to use the keyword `extern`.

5. Is it true that a function may have several declarations, but only one definition?  
   A. Yes  
   B. No  
   **Answer & Explanation**  
   **Answer: Option A**  
   **Explanation:**  
   Yes, but the function declarations must be identical.  
   **Example:**  
   ```c  
   #include<stdio.h>  
   void Display();  
   void Display();  
   void Display();  
   void Display()  
   {  
   printf("Weclose to Saikat12.com..!");  
   }  
   int main()  
   {  
   Display();  
   return 0;  
   }  
   //Output:  
   Weclose to Saikat12.com..!  
   ```
Explanation:

Yes, but the function declarations must be identical.

Example:

```c
#include<stdio.h>

void Display();
void Display();
void Display();

void Display()
{
    printf("Welcome to Saikat12.com..!");
}

int main()
{
    Display();
    return 0;
}

//Output:
Welcome to Saikat12.com..!
```

End