**KEYWORDS, IDENTIFIER, LITERALS, OPERATORS AND EXPRESSION**

**1.** Choose all valid identifiers

a. int int

· Invalid because ‘int’ is a keyword.

b. int \_numvalue

· Valid. It starts with letter and uses an underscore.

c. float price\_money

· Valid. This follows the rule as it starts with a letter and uses an underscore.

d. char name1234567890123456789012345678901234567890

· Valid, but it’s long, it follows the rule.

e. char name value

· Invalid because there is a space.

f. char $name

· Invalid because identifiers cannot start with a special character like’$’.

**2**. What is the meaning of the following keywords, show the usage

a. Auto

· This keyword is used to define automatic variables, which are local to the block in which they are defined.

b. Extern

· This keyword is used to declare avariable or function that is defined in another file or outside the current scope.

c. Volatile

· This keyword is used to indicate that a variable’s value may change at any time without any action being taken by the code the compiler finds nearby.

d. Sizeof

· This operator is used to determine the size of a datatype or variable.

e. Const

· This keyword is used to define a constant variable whose value cannot be change after it is initialized.

**3**. Explain the difference between the following variables.

a. char \*ptr = “ABC”;

· This means that ‘ptr’ points to a location in memory where characters ‘A’,’B’, and’C’ are stored.

b. char arr[]=”ABC”;

· In this case, ‘arr’ is an array of characters that is initialized with the string “ABC”.

Can you manipulate the contents of ptr? Why?

· No, we cannot manipulate the content of ‘ptr’ because it points to a string, which is stored in read-only memory.

Can you manipulate the contents of arr? Why?

· Yes, we can manipulate the contents of ‘arr’ because it is an array that holds a copy of the string “ABC” in writable memory.

Which one of the above is a string literal?

· The variable ‘ptr’ holds a pointer to a string literal “ABC”.

Predict the output of the following code .

void main()

{

//set a and b both equal to 5.

int a=5, b=5;

//Print them and decrementing each time.

//Use postfix mode for a and prefix mode for b.

printf("\n%d %d",a--,--b);

printf("\n%d %d",b++,--b);

}

A: Output:

5 4

4 4

**4.** Refer the code snippet. It fails with error. Fix it.

#include<stdio.h>

int main()

{

int i,k;

const int num;

/\* for(i = 0;i < 9;i++)

{

k = k + 1;

} \*/

num = num + k; /\* Compiler gives the error here \*/

printf("final value of k:%d\n",k);

printf("value of num:%d\n",num);

return 0;

}

A:The line ‘num=num+k;’ has been removed because you cannot modify a constant variable.

**5**. Consider the following code snippet. Evaluate the value of f1, f2 and f3.

int main()

{

int i = 10;

int j = 3;

float f1 = i / j;

float f2 = (float ) i / j;

float f3 = (float ) (i / j);

}

A:

· Here, i is 10 and j is 3.

· This integer result is assigned to a float variable, which will convert it to 3.0

· So, f1=3.0.

f2=(float)i/j:

· The expression becomes ’10.0/3’, which performs floating-point division.

· The result is approximately 3.3333.

F3=(float)(i/j):

· Thus, ‘10/3’ results in 3.

· Then, this integer result(3) is cast to a float, resulting in 3.0.

· So, f3=3.0.