Keywords, Identifier, Literals, Operators and Expression Assignment

Mandatory:

1. Choose all valid identifiers
    a. int int
    b. int _numvalue
    c. float price_money
    d. char name1234567890123456789012345678901234567890
    e. char name value
    f. char $name
    ans:
    a.int int: int is a reserved keyword, and you cannot use it as an identifier.
    e.char name value: This identifier contains a space, which is not allowed in identifiers.

2. What is the meaning of the following keywords, show the usage
    a. auto
    b. extern
    c. volatile
    d. sizeof
    e. const
    ans:
        1. auto:keyword allows the compiler to automatically deduce the type of a variable from its initializer. - auto x = 5;
        2. The extern: declares a variable or function that is defined in another file or outside the current scope. - extern int x;
        3. The volatile :tells the compiler not to optimize a variable, as its value may change unexpectedly - volatile int flag = 0;
        4. The sizeof operator returns the size (in bytes) of a variable or data type. - printf("%zu", sizeof(int));
        5. The const keyword makes a variable's value constant, preventing modification after initialization. - const int MAX_VALUE = 100;

3. Explain the difference between the following variables.
    a. char *ptr = “ABC”;
    ans:
    is a pointer to a string literal
    b. char arr[]=”ABC”;
    ans: is an array initialized with the string literal "ABC".

    Can you manipulate the contents of ptr? Why?

Ans:
You **cannot manipulate** the contents of ptr because string literals are typically stored in read-only memory, while you **can manipulate** the contents of arr because it is a writable array of characters.

Can you manipulate the contents of arr? Why?
Ans: Yes, you can manipulate the contents of arr because it is a character array stored in writable memory, and its elements can be modified.

Which one of the above is a string literal?
Ans:
"ABC" is the string literal, which appears in both **a.** (char *ptr = "ABC";) and **b.** (char arr[] = "ABC";).

4. 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);
}
```

Predicted output:

```
5 4
4 4
```

5. 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;
}
```

program after error fixed:

```
#include <stdio.h>

int main() {
   int i, k = 0;
   const int num = 10;

   for (i = 0; i < 9; i++) {
      k = k + 1;
   }

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

   return 0;
}
```

6. 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);
}
```

Evaluated value:

f1 = 3.0

f2 = 3.333333

f3 = 3.0