**KEYWORDS, LITERALS, IDENTIFIERS, OPERATERS, AND EXPRESSION**

**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**

**a. int int**

* **Invalid:** int is a reserved keyword in C, so it cannot be used as an identifier.

**b. int \_numvalue**

* **Valid**: \_numvalue is a valid identifier because it starts with an underscore, and the name is not a reserved keyword.

**c. float price\_money**

* **Valid:** price\_money is a valid identifier. The name is composed of letters and an underscore, and it doesn't start with a digit or with a reserved keyword.

**d. char name1234567890123456789012345678901234567890**

* **Valid:** This is a valid identifier. It is composed of letters and digits, starting with a letter.

**e. char name value**

* **Invalid:** Identifiers cannot contain spaces. The presence of a space between name and value makes this an invalid identifier.

**f. char $name**

* **Valid:** $name is a valid identifier.

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

**a. auto**

**b. extern**

**c. volatile**

**d. sizeof**

**e. const**

**1. auto**

**Meaning:** The auto keyword is used to declare automatic variables.

**2. extern**

**Meaning:** The extern keyword is used to declare a variable or function that is defined in another file.

**3. volatile**

**Meaning**: The volatile keyword is used to tell the compiler that a variable's value may be changed at any time outside the current code flow.

**4. sizeof**

**Meaning**: The sizeof operator is used to get the size (in bytes) of a data type, variable, or expression.

**5. const**

**Meaning**: The const keyword is used to define a constant variable or constant pointer. A const variable cannot be modified after its initialization.

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

**a. char \*ptr = “ABC”;**

**b. char arr[]=”ABC”;**

a. char \*ptr = "ABC";

This declaration defines ptr as a pointer to char, and it is initialized with the string literal "ABC".

* ptr points to the first character of the string literal "ABC"

**Can you manipulate the contents of ptr? Why?**

No, We cannot manipulate the contents of ptr because it points to a string literal, which is stored in a read-only memory area. Attempting to modify the contents of a string literal is undefined behavior and can lead to a runtime error.

**b. char arr[] = "ABC";**

This declaration defines arr as a character array initialized with the string literal "ABC".

**Can you manipulate the contents of arr? Why?**

Yes, you can manipulate the contents of arr because it is a regular, modifiable character array. The array arr is allocated on the stack or in a writable section of memory, allowing its contents to be modified.

**Which one of the above is a string literal?**

"ABC" is a string literal in both cases

**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);**

**}**

**OUTPUT: 5 4**

**3 4**

**A black background with a black square

Description automatically generated with medium confidence**

**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;**

**}**

**A screenshot of a computer program

Description automatically generated**

**OUTPUT:**

****

**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);**

**}**

**Evaluating f1 = i / j;**

* **Operation**: i / j is the division of two integers (i and j), which results in integer division.
  + Integer division in C truncates the result (it discards the fractional part).
  + i = 10, j = 3, so i / j = 10 / 3 = 3 (result)

**Evaluating f2 = (float) i / j;**

* **Operation:** The cast (float) i makes i a floating-point value before performing the division.
  + i = 10, j = 3, so i is cast to float, which results in 10.0.
  + Now, the division becomes 10.0 / 3, which is floating-point division.
  + 10.0 / 3 equals approximately 3.3333333 (the result is a floating-point value)

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

* **Operation:** The parentheses first perform the integer division i / j before casting the result to float.
  + i = 10, j = 3, so i / j = 10 / 3 = 3 (integer division).
  + The result of i / j (which is 3) is then cast to float, making it 3.0.

**OUTPUT:**

A black background with white numbers

Description automatically generated