# Keywords, Identifier, Literals, Operators and Expression Assignment

**Mandatory:**

1. Choose all valid identifiers
   1. int int - invalid
   2. int \_numvalue - valid
   3. float price\_money - valid
   4. char name1234567890123456789012345678901234567890 - valid
   5. char name value - invalid
   6. char $name - invalid
2. What is the meaning of the following keywords, show the usage
   1. Auto - Automatically deduces the type of the variable.

#include <stdio.h>

int main() {

auto int x = 5;

printf("%d\n", x);

return 0;

}

* 1. Extern - Declares a variable or function that is defined in another file or outside the current scope.

// File 1: main.c

#include <stdio.h>

extern int global\_var

int main() {

printf("Global variable: %d\n", global\_var);

return 0;

}

// File 2: globals.c

#include <stdio.h>

int global\_var = 10;

* 1. Volatile - Tells the compiler not to optimize a variable, as it can be changed unexpectedly

#include <stdio.h>

volatile int flag = 0;

void checkFlag() {

while (flag == 0) {

}

printf("Flag changed!\n");

}

int main() {

checkFlag();

return 0;

}

* 1. Sizeof - Returns the size, in bytes, of a data type or object.

#include <stdio.h>

int main() {

int x = 10;

printf("Size of int: %zu bytes\n", sizeof(x));

printf("Size of int (type): %zu bytes\n", sizeof(int));

return 0;

}

* 1. Const - Defines a constant value that cannot be modified.

#include <stdio.h>

int main() {

const int x = 10

const int\* ptr = &x;

printf("Value of x: %d\n", x);

return 0;

}

1. Explain the difference between the following variables.
   1. char \*ptr = “ABC”;
   2. char arr[]=”ABC”;

Can you manipulate the contents of ptr? Why?

Ans: no we can’t manipulate the contents of ptr because it is read only here.

Can you manipulate the contents of arr? Why?

Ans: yes we can manipulate the contents of array because it is write and can be modified here.

Which one of the above is a string literal?

Ans: both are string literals and the string literal is “ABC”

1. 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

4 4

1. 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;

}

Corrected code:

#include<stdio.h>

int main()

{

int i,k;

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;

}

Output:

Final value of k : 0 (some times it leads to garbage value)

Value of num: 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);

}

f1 = 3.000000

f2 = 3.333333

f3 = 3.000000