Terms	Definition	Example
Bit	Basic unit of information. the value can be **0** or **1**	the character 'A' has the value **01000001** (ASCII value 65)
Byte	A group of 8 bits	char (1 byte), int (4 bytes), float (4 bytes), double (8 bytes), long double (10 or 12 bytes)
Data type	allocated memory space for a value or other information	**int (2 bytes), char (1 byte), float (4 bytes), double (8 bytes)**
Symbolic Constant	Define a constant before compilation using the format: #define *name text*	**#define *course "C-course"***
Preprocessor Directives	# command tells the preprocessor to include header files before the code is compiled	**#include <stdio.h>** **#include <math.h>**
Variable	Memory storage location associated with a name	**int var1;** **char var2;** values can be assigned to variables: **float value = 0.01;**

Variable Scope	Global (outside a function, at the top) or local (within a function)	**int var1;** //global
		int main()
		{
		int var2; //local
		}

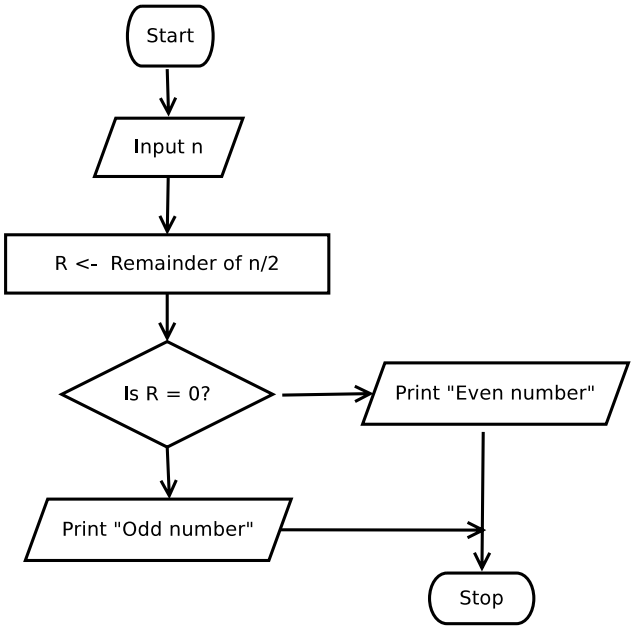
Variable Address	Memory location whose size is measured in 'bytes'. This is where data is stored using the 'address of' operator (&).	int a; scanf("%d", **&**a);
Pointer Variable	Variable that is assigned the address of another C entity. The pointer and C entity must be of the same data type	int number = 9; **int *ptr** = &number
Identifier	Name given to a C entity such as; variable, function, structure etc	**int number;** **char array[10];**
Keyword	Names that are reserved for the compiler. these should **not** be used as variable names	some examples include: **while, else, for, if, sizeof, static, void, short, int, char**
Escape Sequence	Used to format with standard output	**\n** (new line) **\t** (tab) **\0** (end of line) **\a** (bell)
Statement	Perform an action (not evaluated)	printf("hello");
Expression	produces a value	y = x + 1;
Operator	Symbols that works on a value or variable	**+ - * / = % & # ! ? ^ " ' ~ \	: ; . ->** **,** etc
Relational and Logical Operators	operators that require two value, one on either side. Usually part of an **if**, **while** or other conditional statement.	**&& || < <= >= > == !=**
Conditional Operator	Provide control over the path taken by the executed code.	**If, else, for, while, do**
Conditional Statement	Used to control the direction of flow in a program depending on the evaluation of the condition	**if, else if, else** if(condition) {…};

Function	Block of code that has a specific job. It has a prototype, declared above main and a definition block	**int function_name(parameters);** **{** **block of code;** **return value;** **}**
Function Header / Prototype	A function is declared before main()	**int function_name(parameters);**
Function Definition	This includes the function header and the block of code associated with it.	**int function_name(parameters);** **{** **block of code;** **return value;** **}**
Parameter	Datatype used in the function prototype and definition	
Argument	A value passed to the function to be processed.	

Spot the mistakes in these two pieces of code - most are **syntax** errors and there are a couple of **logical** errors.

```
#include <stdio.h>

int main() {
    char input[];
    printf("enter word: \n");
    scanf("%c", input);

    for(int i=0; i<sizeof(input); i++)
    {
        printf("%c", input[i]);
    }

    return 0;
}
```

```
#include <stdio.h>

int main()
{
    int x,i;
    do
    {
        printf("Please give an integer from 1 to 20 : ");
        scanf(“%d ”, &x);
     }
     While( x>20 or x<1 );

    for (i=1; i>=x; i+=1)
    {
         printf("%d\n",i);
    }
    return(0);
}
```



C

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    int n, R;
    scanf("%d", &n);

    R = n%2;
    if(R == 0)
    {
        printf("Even number\n");
    }
    else
    {
        printf("Odd number\n");
    }
    return 0;
}
```

Python

```python
n = int(raw_input())
R = n%2
if(R==0):
    print("Even number")
else:
    print("Odd number")
```

BASIC

```
input " ", n
R = n%2
if R=0 then print "Even number"
else print "Odd number"
end if
```

R

```r
n<-as.numeric(readline())
R<-n%%2
ifelse(R==0, print("Even number"), print("Odd number"))
```

Scheme (Racket)

```
(define n (read))
(define R n)
(if (equal? (modulo R 2) 0)
    (display "Even number")
    (display "Odd number")
)
```