Session A 06

Tuesday, 16 September 2025 9:22 AM

(4)

+ve

0000
0001
0010
0011
0100
0101
0110
0111

-ve

0000 (-1)
1111 (-2)
1110 (-3)
1101 (-4)
1100 (-5)
1011 (-6)
1010 (-7)
1001

$-1\times2^3 + 0\times2^2 + 0\times2^1 + 0\times2^0$

(-8)

1000

-7 to +7

-8 to +7

(1010)

$+8 + (-7) = 1$

$+8 - (+7)$

2's Complement is a weight system —

But Sign Bit always treated as (-ve) number

0111

$-0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

7

1001

$-1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$

$-8 + 0 + 0 + 1 \Rightarrow -7$

$+8 \rightarrow$

$\Rightarrow$

1 1 → Different

+8: 0 0 0 0 1 0 1 0

→: 1 1 1 1 1 0 0 1

0 0 0 0 0 0 0 1

+ve

1000

$-1\times2^3 + 0\times2^2 + 0\times2^1 + 0\times2^0$

-8

20 = 10 1 0

5 = 1 0 1

0 0 0 0 1 0 1 0

1 1 1 1 0 1 1 0

int x = 20;

int y = -11;

int c = x & y;

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

&

x →

+20: 0 0 0 1 0 1 0 0

-11: 1 1 1 1 0 1 0 1

0 0 0 1 0 1 0 0

C ← 20

XOR →

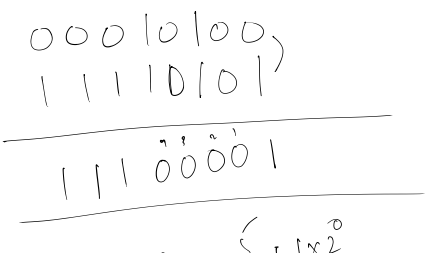
Truth Table

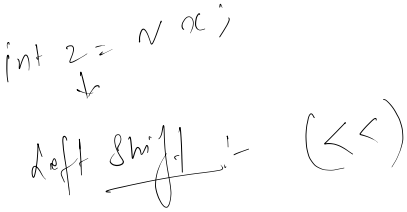
A	B	Y
0	0	0
0	1	1
1	0	1

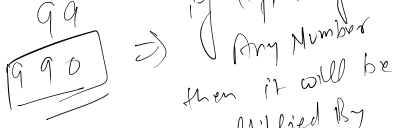
Truth Statement

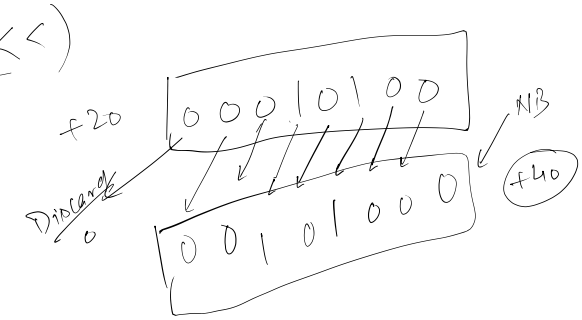
If Input are same output is zero

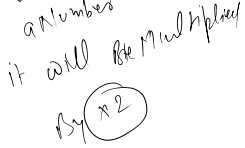
Otherwise one

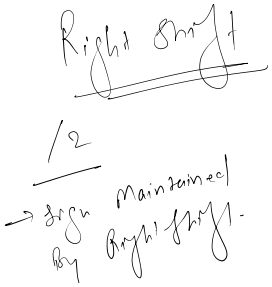


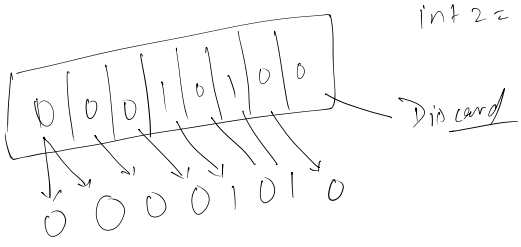


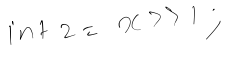












In C programming, **storage classes** define the **scope**, **visibility**, **lifetime**, and **default initial value** of variables and functions. They determine where a variable or function can be accessed from and how long it exists in memory.
There are **four storage classes** in C:

1. auto

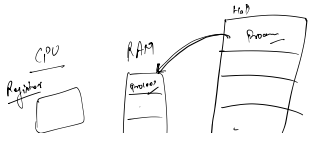
- **Default** storage class for local variables.
- Automatically assigned to variables declared **inside a function** if no other storage class is specified.
- **Scope**: Local to the block in which it's defined.
- **Lifetime**: Exists during the execution of the block.
- **Default initial value**: **Garbage value** (uninitialized).

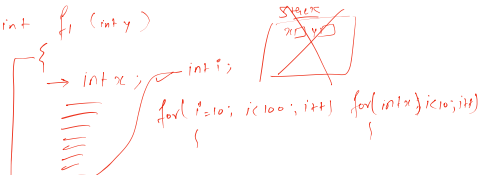
```
void example() {
  auto int x = 5; // 'auto' is optional here
  printf("%d", x);
}
```

2. register

- Suggests that the variable be stored in a **CPU register** instead of RAM for **faster access**.
- **Scope**: Local to the block.
- **Lifetime**: Duration of the block.
- **Default initial value**: **Garbage value.**
- You **cannot get the address** of a register variable using the & operator.

```
void example() {
  register int speed = 20;
  printf("%d", speed);
}
```





3. static

- Keeps the value of a variable **between function calls**.
- Can be used with **local variables** and **global variables/functions**.
- For local variables:
  - **Scope**: Local to the function/block.
  - **Lifetime**: Entire program execution.
  - **Default initial value**: **Zero**.
- For global variables/functions:
  - **Visibility**: Limited to the file (file scope).

```
void example() {
  static int count = 0;
  count++;
  printf("%d", count);
}
// Output across multiple calls: 1 2 3 ...
```

4. extern

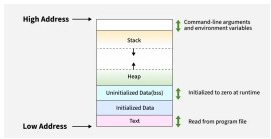
- Used to **declare a global variable or function** that is defined in another file or later in the code.
- **Scope**: Global.
- **Lifetime**: Entire program.
- It does **not allocate memory**; just declares the variable.

```
// file1.c
int count = 10;
// file2.c
extern int count;
void printCount() {
  printf("%d", count);
}
```

Summary Table

Storage Class	Scope	Lifetime	Default Value	Usage
auto	Local	Block	Garbage	Default for local variables
register	Local	Block	Garbage	Fast-access (no address)
static	Local/File	Entire program	Zero	Persistent local vars
extern	Global	Entire program	Zero (if defined)	Access global vars/functions

If you want a deep dive into any one of them or example use cases, let me know!




```
#include <stdio.h>
void Hello()
{
    static int x = 10;
    printf("%d\n", x);
    x++;
}
int main()
{
    Hello();
    Hello();
    Hello();
    Hello();
    Hello();
}
```

```
int fib(int n){
}
```

```
int fib(int n) {
    int no = 0;
    int n1 = 1;
    int nextterm;
    if(n<=1)
    return n;
    while(i<=n)
    {
    nextterm = no+n1;
    no = n1;
    n1 = nextterm;
    i++;
    }
    return nextterm;
}
```

```
int arr[10];
datatype name[size];
```

```
int arr[5] = {1,2,3,4,5};
```