Switch & functions

* Switch Case:

Syntax:

```
switch (x) {
    case y:
        ~~~
        break;
    case 2:
        ~~~
        break;
    default:
        ~~~
}
```

switch statement compare x with y if it's true then execute y's code then we write break. because after that x should not compare with 2. And if you want to compare x with other case then don't use break.

→ there is also default case. which will execute if there is no case match

→ default is not mandatory.

→ we can also make nested switch.

→ exit ( ) :

→ continue statement in switch case is not valid or not useful.

Q. Make simple calculator.

Q. Amount will given. How many no. of ₹100 notes ₹50 notes ₹20 notes and ₹1 notes we will need.

i/p: 1380

Q. power (i/p a=2, b=3, ans: $2^3=8$)

*functions: → do well defined task only.

→ code become readable
→ code become less bulky
→ there is less possibility of bugs.

Syntax:

```
return_type function_name (argument1, argument2...){

    executable code ----

    return value;

}
```

Q. power of any number

```
int power(a,b){
    int ans=1;
    for(int i=1; i<=b; i++){
        ans=ans*a;
    }
    return ans;
}
int main(){
    int c=power(2,3);
    cout<<c<<endl;
}
```

→ 8.

Q. Number is even or odd?

Q. find ^nC_r ($^nC_r = \frac{n!}{r!(n-r)!}$)

Q. print counting 1 to n

→ If there is no return value then return_type = void.

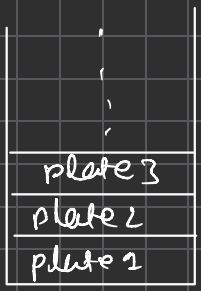
Q Number is prime or not.

* Function call stack:

Q. ^nC_r question there are 3 fns fact(), nCr(), main()

Stack:

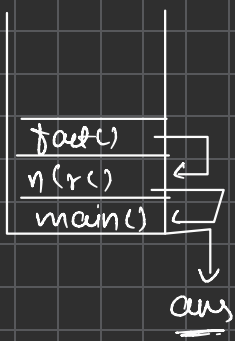
ex: plates, chairs on one another



→ LIFO (last in first out)

↳ The plate we insert last will be on top that will first out when we are picking the plates

↳ function call stack:



Q: A.P = (3×n+7) find n^{th} term of this series.

Q: i/p → a, b , o/p → total number of set bits in a & b.

Q: Fibonacii series's n^{th} term.

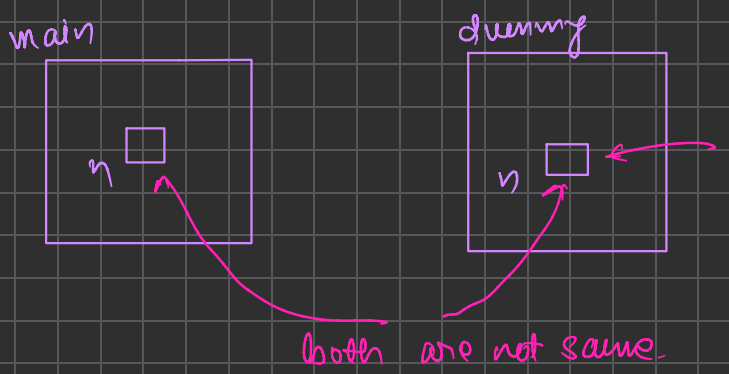
→ Note: return 0; is for to make function call stack empty.

* Pass by value:

```
void dummy(int n){
    n++;
    cout << "n is" << n << endl;
}

int main(){
    int n;
    cin >> n;
    dummy(n);
    cout << "n is" << n << endl;
}
```

Output: n is 16 ← dummy
n is 15 ← main.



→ if we increment this n then it won't effect on n which is main.

→ name of variables are same but address are different.

See the output question at 1:22:53 onwards.