for function

Q. find is number is power of 2!

↳ Approach 1:

```
while(n>=1){
    if(n%2==0){
        n=n/2;
    }
    else{
        cout<<"not power of 2"; break;
    }
}
```

Approach 2: find number of set bits

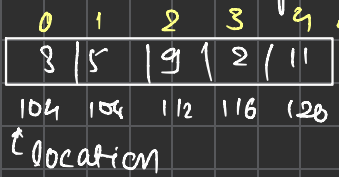
↳ if setbits=1 then Yes

else No

* Arrays *

→ we can store values of same data types.

→ store values in contiguous location.



→ can access the value by index.

Declaration: data_type array_name [size_of_array];

int dost [10];

→ array_name also represents the address of first element.

dost [0] = 5 → dost + size × index = 100 + 4 × 0 = 100 at 100 there is 5.

dost [6] = 4 → dost + size × index = 100 + 4 × 6 = 124 at 124 there is 4.

→ Initialize: int arr [3] = {5, 7, 11};

int arr [10000] = {0}; int arr [10000] = {};

int arr [10000];

int arr[1000] = {1}; → This won't work

int arr[1000] = {[0...999] = 1}; → This will work

↳ else we can do it with the help of for loop.

→ If we will access the element more than size then it will show error: Array index out of bounds

* Note: Array always pass by reference in function.

Reason:

```
void fn(int arr[]){

}

int main(){
    int arr[10] = {5, 3, 7, ....};
    fn(arr)
}
```

↑ here we pass arr which denotes the first location of that array not value

* length of array:

①
```
for(auto i : arr){
    count++;
}
```

② end(arr) - begin(arr)

③ sizeof(arr)/sizeof(data_type);

④ arr.size();

* bad practice:

```
int size;
cin >> size;
int arr[size];
```

} instead of this `int arr[1000000];` ← make very big value.

* input in array:

```
int size;
int arr[10000];
cin >> size;
for(int i=0; i<size ; i++){
    cin >> arr[i];
}
```

* INT_MIN , INT_MAX :

INT_MIN ↑ min value of integer

INT_MAX ↑ max value of integer.

Q. get max value from array.

Q. get min value of array.

Q. print sum of elements in array.

* linear Search:

Q.
```
int arr[10] = {5, 7, -2, 10, 22, -2, 0, 5, 22, 1};
find key: 0 !
for(int i: 0; i<size, i++){
    if (key== arr[i]){
        cout << "found" <<endl;
        break;
    }
}
cout << "element not present" <<endl;
```

Q. Reverse An Array:

```
for(int i:0, i<size/2 ; i++){
    swap(arr[i], arr[size-i-1]);
```

y

Q. Swap Alternate: i/p ⇒ { 1, 2, 3, 4, 5, 6 }

o/p ⇒ { 2, 1, 4, 3, 6, 5 }

Q. find unique in Array.

Q. find duplicate in Array.

Q. Intersection of Array.

Q. Pair sum

Q. Triplet sum

Q. Sort 0's and 1's

Soln in next lecture notes or codes.