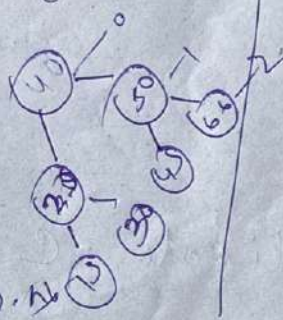
DSA

2nd Sem

Linear Search.

1) Wap in C to perform a linear search to find a specific (key) item in an array.

Ans) Analogy:- We have to find the item from the array.



0	1	2	3	4	5
1	2	3	4	5	6

size = 6

key = 3 (2) index

Code :-

```
#include <stdio.h>
int main(){
    int a[5] = {1,2,3,4,5};
Reverse(a, 5, 3); int result =
Print(a,5,
Reverse(a,5,3);
if (Reverse(a,5,3) != -1){
```

~~Printf ("key %d was been found");~~

```
if (result != -1) {
        Printf ("%d key : result);
}
else printf ("Not found");
void Reverse (int a[], int size, int key)
{
    int i = 0;
    int j = size - 1;
for (i=0; i<=size-1; i++) {
    if (a[i] == key) {
        return i;
    }
return -1;
}
```

2) Duplicate .c :-

Code

```
#include <stdio.h>
int main () {
int a[5] = {1, 2, 3, 1, 2};   // 0 1 2 3 4
int c = 0;
int i, j;
```

```
for (i=0; i<n; i++){
    for (j=i+1; j<5; j++){
        if (a[i] == a[j]){
            c++;
        }
    }
}
printf ("The duplicate elements are %d", c);
return 0;
```

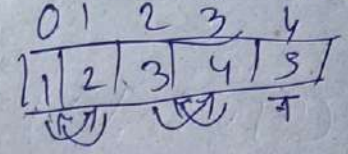
3. Reverse. C

	0	1	2	3	4
I/P	1	2	3	4	5
	0	1	2	3	4
O/P	5	4	3	2	1

approach 1-

```
#include <stdio.h>
int main(){
int i, a[5] = {1,2,3,4,5};
for (i=5; i>0; i--){
    printf ("%d", a[i]);
}
return 0;
}
```

approach 2:- In order to reverse the array we can swap the corresponding elements ex:-



Implementation:-

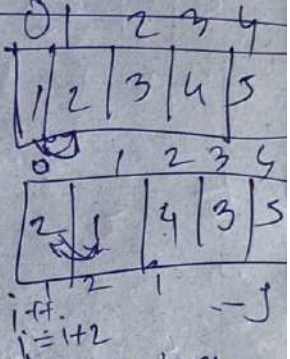
```
#include <stdio.h>
void reverse (int a[], int size) {
    int i=0;
    int j= size -1;
    while (i<=j) {
        swap (a[i], a[j]) {
            i++;
            j--;
        }
    }
void print (int a[], int size) {
    int i;
    for (int i=0; i<=size-1; i++) {
        printf ("%d", a[i]);
    }
}
```

```
int main (){
  int n;
printf ("Enter the number/size n");
  Scanf ("%d", &n);
    int a[n];
reverse (a, n);
print (a, n);
return 0;
}
```

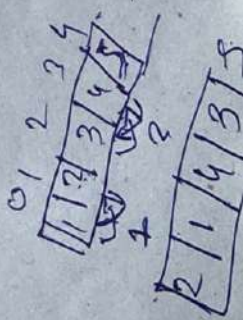
4) Swap alternatives.c

Analogy:- Swap the corresponding elements in an ~~array~~ array.

```
#include <stdio.h>
void swap (int a[], int size){
  ~~int~~ i
for (int i=0; i<size; ~~i++~~ i+=2){
    if (a[i+1]<size){
      ~~int~~ int t = a[i];
        a[i] = a[i+1];
        a[i+1] = t;
    }
}
```



if (i) was done swapping the 2,1 element i+=2 / i=i+2



```
void print (int a[], int size) {
    for (int i=0; i<size; i++) {
        printf("%d", a[i]);
    }
}

int main () {
    int a[5] = {1,2,3,4,5};
    swap (a, 5);
    print (a, 5);
    return 0;
}
```

5) Swap the alternative of an non increasing array?

#

given:-

0	1	2	3	4
6	5	4	3	-2

0	1	2	3	4
-2	3	4	5	6

0	1	2	3	4
3	-2	5	4	6

```
#include <stdio.h>
void sort (int *a, int size) {
    for (int i=0; i<=size-1; i++) {
        for (int j=i+1; j<=size-1; j++) {
            int t = a[i];
            a[i] = a[j];
            a[j] = t;
        }
    }
}

void swap (int *a, int size) {
    for (int i=0; i<size; i=i+2) {
        if (a[i+1] < size) {
            int t = a[i];
            a[i] = a[i+2];
            a[i+1] = t;
        }
    }
}
```

```
void print (int *a, int size) {
    for (int i=0; i<=size-1; i++) {
        printf ("%d", a[i]);
    }
}

int main () {
    int a[6] = {6, 5, 4, 3, 2, -1};
    int size = sizeof (a) / sizeof (a[0]);
    Sort (a, size); // for sorted array
    swap (a, size); // for swapping func
    print (a, size); // for printing ...
    return 0;
}
```

6) // Find a pairs in an array using linear Search

example:

0	1	2	3	4	5
1	2	3	4	5	6

x = 9

The pair is one - (4+5) = 9

```
#include <stdio.h>
int main() {
    int a[5] = {1, 2, 3, 4, 5};
    int size = sizeof(a)/sizeof(a[0]);
    int i, j;
int x;
printf("Enter the x ");
    scanf("%d", &x);
for (i=0; i<size; i++) {
    printf("%d\t", a[i]);
    }
printf("\n");
int c = 0;
for (i=0; i<size; i++) {
    for (j=i+1; j<size; j++) {
        if (a[i] + a[j] == x) {
            c++;
    }
}
}
printf("The pair of %d is %d", c);
    return 0;
}
```

7) Find the triplets of any elements in the array.

Example

0	1	2	3	4	5	6
1	2	3	4	5	6	7

x = 6 (3+2+1)

Code:-

```
#include <stdio.h>
int main () {
int arr[10] = {1,2,3,4,5,6,7,8,9,10};
int size = sizeof (arr) / sizeof (arr[0]);
int i, j, k;
for (i=0; i<size; i++) {
    Printf ("%d", a[i]);
}
Printf ("\n");
Printf ("Enter the x for finding triplets");
Scanf ("%d", &x);
```

```
int c=0;
for (i=0; i<size; i++){
    for (j=i+1; j<size; j++){
        for (k=j+1; k<size; k++){
            if (a[i]+a[j]+a[k]==n){
                c++;
            }
        }
    }
}
printf("%d triplets are %d", c);
return 0;
}
```

8) Find the Specific element (x) and return the index of that element & sum the whole array.

```
#include <stdio.h>
int main(){
int a[10] = {1,2,3,4,5,6,7,8,9,10};
int size = sizeof(a)/sizeof(a[0]);
int x;
printf("Enter the n");
scanf("%d", &n);
```

```
(8) for (int i=0; i<size; i++) {
(9)     printf("%d\t", a[i]);
    }
(10) Scanf int tot = search(a, size, n);
(11) if (tot != -1) {
(12)     printf("%d", tot);
    }
(13) else printf("Not found");
(14) int sum = 0;
(15) for (i=0; i<size; i++) {
(16)     sum += a[i];
    }
(17) printf("%d", sum);
(18) return 0;
(19) }
(20) int search (int *a, int size, int x
     key) {
(21)     int i;
(22)     for (i=0; i<size; i++) {
(23)         if (a[i] == x) {
(24)             return i;
(25)         }
(26)     }
(27)     return -1;
(28) }
```

Binary Search

In this Search technique we divide an array into two phases and find a mid point. if the ~~search~~ in required element is ~~right~~ greater than the mid index we go right else we traverse left if the element gets found we simply return the index else we return the -1.

→ Sorted array

0	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9

4

0	1	2	3
1	2	3	4

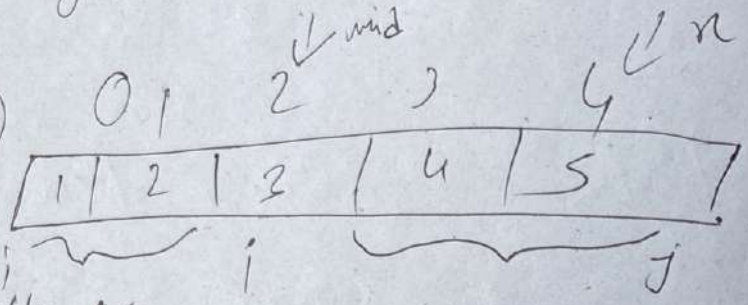
mid

5	6	7	8	9
5	6	7	8	9

$\frac{0+9}{2} : \frac{9}{2} : 4$

The formula is $\frac{start + end}{2}$

1) Implement a code where we find x from an sorted array using "Binary Search" Algorithm.



$\frac{0+4}{2} = \frac{4}{2} = 2$

$\frac{2+4}{2} = \frac{6}{2}$

x = 4

Pass ① if a[mid] == x ✗

2 == 4 ✗

else if a[mid] > x

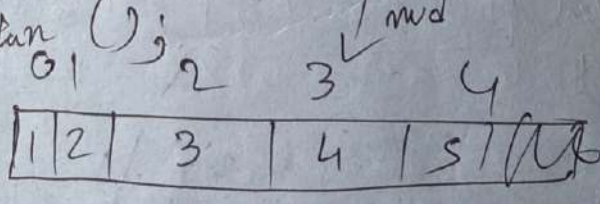
2 > 4 ✗

else a[mid] < x ✓

2 < 4 ✓

i = mid + 1

fun ();

② 

$\frac{2+4}{2} = \frac{6}{2} = 3$

i

3 < 4 ✓

Pass ② a[mid] == x ✗

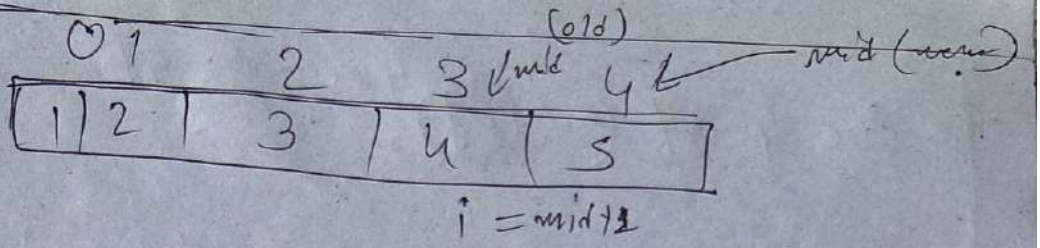
3 == 4 ✗

a[mid] > x

3 > 4 ✗

i = mid + 1

fun ();



Pass (3)

if a[mid] == x (4 == 4)

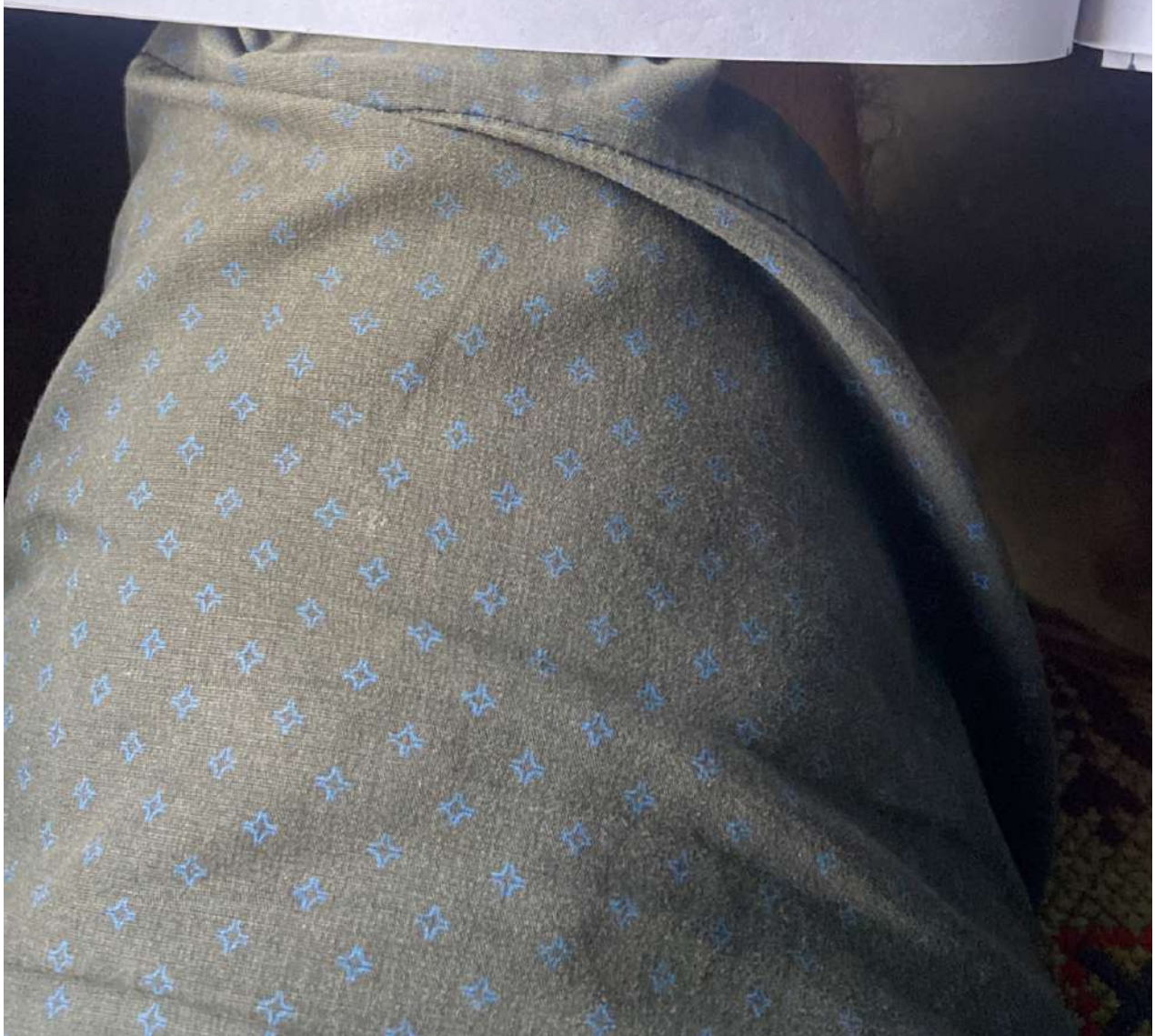
return mid; ← found.

Let's Implement the code :-

```
#include <stdio.h>
int main () {
    int a[10] = {1,2,3,4,5,6,7,8,8};
    int size = sizeof (a)/sizeof (a[0]);
    int i,j,x;
    printf ("Enter the x");
    scanf ("%d", &x)
    bin_search ( );
    int y = -1;
    y = bin_search (a, i, j, x);
    if (y == -1) {
        printf ("Element Not found");
    }
    3
```

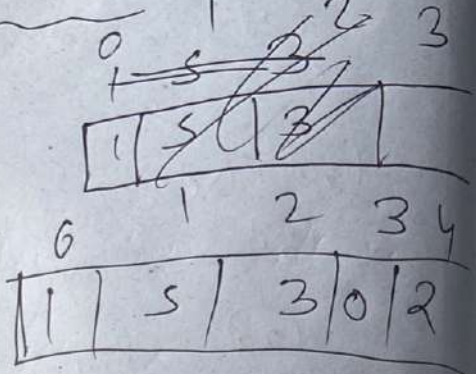
```
else printf("%d", y);

⑥ for (i=0; i<size; i++){
    ⑦ printf("%d\t", a[i]);
    ⑧ }
⑨ printf("\n");

⑰ return 0;
⑱ }

⑲ int bin_Search(int *a, int i, int j, int x){
    ⑳ int mid = (i+j)/2;
    ㉑ if (a[mid] == x){
        ㉒ return mid;
        }
    ㉓ if (i == j){
        ㉔ return -1;
        ㉕ }
    ㉖ else if (a[mid] > x){
```

```
int bin_Search (int *a, int i, int j, int x)
{
    int mid = (i+j)/2;
    if (a[mid] == x)
    {
        return mid;
    }
    if (i == j)
    {
        return -1;
    }
    else if (a[mid] > x)
    {
```

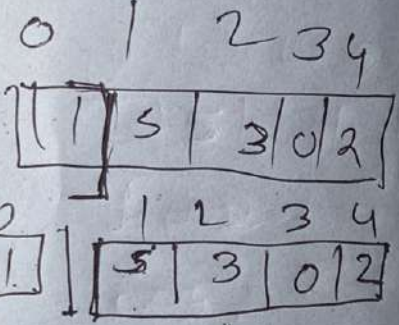


```
27  i=mid+1; j=mid;
28  bin_Search(a,i,j,x);
29  }
30  else {
31  j=mid; i=mid+1;
32  bin_Search(a,i,j,x);
33  }
34  }
```

Insertion Sort

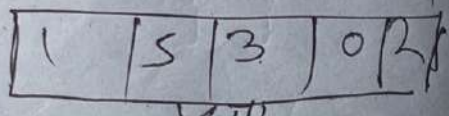


Analogy:- We have to first assume that the array's one part is sorted and another one is unsorted part. Let's see an example



This part ← is sorted and (n-1) is unsorted

for ← 0 1 2 3 4



J-1 a[J]

if a[J] < a[J-1]

3 < 5

Swap

J--;

Condition 1) If a[j] < a[j-1]
swap;
j--;

2) if j>0 or j<=1
Traverse the loop
unless j>0 or j>=1

0	1	2	3	4
1	5	3	0	2

a[j]
a[j-1]

if a[j] is at j'th index and a[j-1] is at 0 index and the condition the still true and if the j gets "--" and after 0'th index if the "j--" happens the loop will never be broken

so we have to

give a condition which is if (j>=1)
let the loop run till j gets 1 or 0 and not -1 or -2 index:

if (j>=1 && a[j] < a[j-1]) {

swap;
}

3 < 5

2) Insertion Sort

P (2) (array negative)

```
#include <stdio.h>
void sort (int *a, int size) {
    int i, j, key;
    for (i=1; i<size; i++) {
        key = a[i];
        j = i-1;
        while (j>=0 && a[j]>key) {
            a[j+1] = a[j];
            j--;
        }
        a[j+1] = key;
    }
}
```

```
void print (int *a, int size) {
    int i;
    for (i=0; i<size; i++) {
        printf ("%d\t", a[i]);
    }
}

int main () {
    int a[10] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
    int size = sizeof (a) / sizeof (a[0]);
    sort (a, 10);
    print (a, 10);
    return 0;
}
```

Selection Sort

(always positive and select)

```
#include <stdio.h>

void swap (int *a, int *b)
{
int temp = *a;
*a = *b;
*b = temp;
}
```

algo ①

```
void sort (int *a, int size)
{
int i,j;
for (i=0; i<size; i++)
{
for (j=i+1; j<size; j++)
{
if (a[i]>a[j])
{
swap (&a[i], &a[j]);
}
}
}
}
```

algo 2

```
void Sort (int *a, int size){
    int i, j, minder;
    for (i=0; i<size; i++){
        minder = i;
        for (j=i+1; j<size; j++){
            if (a[j] < a[minder]){
                minder = j;
            }
        }
    }
    swap (&a[i], &a[minder]);
    }
}

Void Print (int a[], int size){
    int i;
    for (i=0; i<size; i++){
        Printf ("%d\t", a[i]);
    }
}
```

```
int main () {
    int a[] = {1, 2, 23, 24, 42, 45, 64, 340,
               -4, 40};
    int size = sizeof (a) / sizeof (a[0]);
    sort (a, size);
    print (a, size);
    return 0;
}
```

Bubble.sort.C

```
#include <stdio.h>
#include <stdbool.h>

void swap (int *xp, int *yp)
{
    int t = *xp;
    *xp = *yp;
    *yp = t;
}

void sort (int a, int size)
{
    int i, j;
    bool swapped;
```

```
for (i=0; i<size; i++){
    swapped = false;
    for (j=0; j<size-i-1; j++){
        if (a[j] > a[j+1]){
            swap(&a[j], &a[j+1]);
            swapped = true;
        }
    }
    if (swapped == false){
        break;
    }
}

void print (int a[], int size){
    for (i=0; i<size; i++){
        Printf("%d\t", a[i]);
    }
}

int main(){
    int a[] = {10, 9, -9, 90, 5, 39, 6
    int size = sizeof(a) / sizeof(a[0]
```

```
sort (a, size);
Print (a, size);
    return 0;
}
```

example eg [10, 9, 8, 70, 6,]?

~~P-1~~ P①

9 10 8 70 65

9 8 10 70 65

~~9 8 10 6 5 70~~

P-②

9 8 10 65 70

P-③

8 9 10 65 70

The big bubble / element sugos unless it gets to the ~~end~~ end and each passes will deliver the greatest, second greatest, third greatest, n greatest element to the ~~end~~ end unless it gets sorted