Bubble Sorting

Date: _____

Array [11, 1, 5, 4, 3, 2]

size = 6

outer loop = size − 1 = 6 − 1 = (5)

inner loop = size − i − 1 = 6 − i − 1

1st iter	2nd iter	3rd iter	4th iter	5th iter
11, 1, 5, 4, 3, 2	1, 5, 4, 3, 2, 11	1, 4, 3, 2, 5, 11	1, 3, 2, 4, 5, 11	1, 2, 3, 4, 5, 11
1, 11, 5, 4, 3, 2	1, 5, 4, 3, 2, 11	1, 4, 3, 2, 5, 11	1, 3, 2, 4, 5, 11	1, 2, 3, 4, 5, 11
1, 5, 11, 4, 3, 2	1, 4, 5, 3, 2, 11	1, 3, 4, 2, 5, 11	1, 2, 3, 4, 5, 11	↓ final array
1, 5, 4, 11, 3, 2	1, 4, 3, 5, 2, 11	1, 3, 2, 4, 5, 11		
1, 5, 4, 3, 11, 2	1, 4, 3, 2, 5, 11			
1, 5, 4, 3, 2, 11				
i = 0	i = 1	i = 2	i = 3	i = 4

```
For (int i=0; i < n-1 , i++) {        n = 6 - 1
    for (int j=0 ; j < n-i-1 ; j++) {
        if (A[j] > A[j+1])
        swap (A[j], A[j+1]);
    }
}
```

Code:

```
#include <iostream>
using namespace std;
Void Bubble_Sort ( int array[ ], int size) {
for (int i = 0 ; i < size-1 ; i++) {
for ( int j = 0 ; j < size-i-1 ; j++) {
if (array [j] > array [j+1])
```

```
        swap (array [j], array [j+1]); }
    }
}

Void display_array(int array[ ], int size){
  for (int i=0 ; i<n ; i++) {
  cout << array [i] << "  "; }
    cout << endl;
}
  int main ( ) {
  int size = 6;
  int array [ ]= {11, 1, 5, 4, 3, 2};
  cout <<"original array" <<endl;
  display_array (array, size);
   Bubble_sort (array, size);
cout <<" Sorted array" << endl;
  display_array (array, size);
   return 0;
}
```

Bubbe Sorting

array [18, 12, 9, 20, 15, 10, 8]

size = 7;

outerloop = size - 1 = 7 - 1 = (6)

innerloop = size - i - 1 = 7 - i - 1

Date: _____

1st iter	2nd iter	3rd iter	4th iter
18, 12, 9, 20, 15, 10, 8	12, 9, 18, 15, 10, 8, 20	9, 12, 15, 10, 8, 18, 20	9, 12, 10, 8, 15, 18, 20
12, 18, 9, 20, 15, 10, 8	9, 12, 18, 15, 10, 8, 20	9, 12, 15, 10, 8, 18, 20	9, 12, 10, 8, 15, 18, 20
12, 9, 18, 20, 15, 10, 8	9, 12, 18, 15, 10, 8, 20	9, 12, 15, 10, 8, 18, 20	9, 10, 12, 8, 15, 18, 20
12, 9, 18, 20, 15, 10, 8	9, 12, 15, 18, 10, 8, 20	9, 12, 10, 15, 8, 18, 20	9, 10, 8, 12, 15, 18, 20
12, 9, 18, 15, 20, 10, 8	9, 12, 15, 10, 18, 8, 20	9, 12, 10, 8, 15, 18, 20	
12, 9, 18, 15, 20, 20, 8	9, 12, 15, 10, 8, 18, 20		
12, 9, 18, 15, 10, 8, 20			

sorted (1st iter) | sorted (2nd iter)

i=0	i=1	i=2	i=3
j = n-i-1	n-i-1	n-i-1	n-i-1
7-0-1 =	7-1-1	7-2-1	7-3-1
6	5	4	3

5th iter	6th
9, 10, 8, 12, 15, 18, 20	9, 8, 10, 12, 15, 18, 20
9, 10, 8, 12, 15, 18, 20	8, 9, 10, 12, 15, 18, 20
9, 8, 10, 12, 15, 18, 20	

Sorted array

i=4	i=5
n-i-1	n-i-1
7-4-1	7-5-1
2	1

Code:

```
#include <iostream>
using namespace std;
```

Date: _____

```
Void Bubble_Sort(int array[ ], int n) {
  for (int i=0 ; i<n-1 ; i++) {
    for (int j=0 ; j=n-i-1 ; j++) {
      if (array[j] > array[j+1])
        swap (array[j], array[j+1]); }
  }
}

Void print_array (int arr[ ], int n){
for (int i=0 ; i<n ; i++){
cout << arr[i] << "  "; }
cout << endl;
}

int main ( ) {
  int size = 7
  int array [ ] = { 18, 12, 9, 20, 15, 10, 8};
cout << "original array" << endl;
Print_array (arr, n);
Bubble_sort (array, n);
cout << "sorted array" << endl;
Print_array (arr, n);
return 0;
}
```

Date:

Selection Sorting

① Array {11, 1, 5, 4, 3, 2}

n or size = 6

outer loop = n-1 time run → 6 → 6-1 = (5) iterations of outer loop

inner loop = (i+1) → n/size

At initial to find smallest element (n-1) → iteration → Push the smallest element to forward.

total array is divided into two parts one is sorted and other is unsorted part.
Firstly, the 1st element assumes to be minimum element of array then compare the remaining elements one by one.
If any one is less than that min element then swap the new min elemt to current min by right Positio

1st iter	2nd iter	3rd iter
11, 1, 5, 4, 3, 2	1, 11, 5, 4, 3, 2	1, 2, 5, 4, 3, 11
↓ assumes to be 1st elemt as min elemt	sorted / unsorted	sorted / unsorted
	i = 1	i = 2
	but j loop start → i+1 elemt	j → start from i+1 elemt
11, 1, 5, 4, 3, 2	1, 11, 5, 4, 3, 2	1, 2, 5, 4, 3, 11
	↓ assumes to be min	↓ assumes to be min
11, 1, 5, 4, 3, 2	1, 11, 5, 4, 3, 2	1, 2, 5, 4, 3, 11
11, 1, 5, 4, 3, 2	1, 11, 5, 4, 3, 2	1, 2, 5, 4, 3, 11
11, 1, 5, 4, 3, 2	1, 11, 5, 4, 3, 2	1, 2, 5, 4, 3, 11
11, 1, 5, 4, 3, 2	1, 11, 5, 4, 3, 2	1, 2, 3, 4, 5, 11
swapping occur because new min is lesser than current min	1, 11, 5, 4, 3, 2	
1, 11, 5, 4, 3, 2	1, 2, 5, 4, 3, 11	
i = 0		

Date: _____

Code:

```
#include <iosteam>
using namespace std;
void selection_sort (int arr [ ], int size){
for(int i=0 ; i< n-1 ; i++){
for int smallest_Index = i;
for (int j= i+1 ;j< n ; j++){
    if (arr [j] < arr [smallest_Index])
    smallest_index = j;
}
swap (arr [i] , arr[smallest_index])
}
void print_array (int arr[ ], int size){
for (int i=0; i<size ; i++)
   cout << arr [i] <<"    ";
cout << endl;
}
   int main ( ){
   int size = 6;
int array [ ] = {11, 1, 5, 4, 3, 2};
print_array (arr,size);
selection_sort (arr, size);
print-array (arr, size);
return 0;
}
```

Date:

Selection sorting

② Array [18, 12, 9, 20, 15, 10, 8]

n / size = 7

outer loop = n-1 = 7-1 = 6

inner loop = initialize from (i+1) → to (n)

1st iter

i=0

~~inner loop i++ = 0++ = 1~~

18, 12, 9, 20, 15, 10, 8

assumes as min

comparing ~~the other elem with~~ current min

18, 12, 9, 20, 15, 10, 8

18, 12, 9, 20, 15, 10, 8

18, 12, 9, 20, 15, 10, 8

18, 12, 9, 20, 15, 10, 8

18, 12, 9, 20, 15, 10, 8

8, 12, 9, 20, 15, 10, 18

2nd iteration

i=1

~~j start from i+1 = 1+1 = 2~~

8, 12, 9, 20, 15, 10, 18

sorted | unsorted

assumes to be current min

8, 12, 9, 20, 15, 10, 18

8, 12, 9, 20, 15, 10, 18

8, 12, 9, 20, 15, 10, 18

8, 12, 9, 20, 15, 10, 18

8, 9, 12, 20, 15, 10, 18

3rd iter

i=2

8, 9, 12, 20, 25, 10, 18

sorted | unsorted

8, 9, 12, 20, 15, 10, 18

8, 9, 12, 20, 15, 10, 18

8, 9, 12, 20, 15, 10, 18

8, 9, 12, 20, 15, 10, 18

8, 9, 10, 20, 15, 12, 18

4th iter

i=3

8, 9, 10, 20, 15, 12, 18

sorted | unsorted

8, 9, 10, 20, 15, 12, 18

8, 9, 10, 20, 15, 12, 18

8, 9, 10, 20, 15, 12, 18

8, 9, 10, 12, 15, 20, 18

Date:

5th iteration

i = 4

j = 4+1 = 5

8, 9, 10, 12, 15, 20, 18

sorted | unsorted

8, 8, 10, 12, 15, 20, 18

8, 9, 10, 12, 15, 20, 18

no swapping occur because the current min is not lesser than any remaining element.

6th iteration

i = 5

j = 5+1 = 6

8, 9, 10, 12, 15, 20, 18

sorted | unsorted

8, 9, 10, 12, 15, 20, 18

8, 9, 10, 12, 15, 18, 20

Final array.

Code:

```
#include <iostream>
using namespace std;
void selection(int arr[ ], int n){
  for(int i=0 ; i<n-1 ; i++){
  int smallest = i ;}
for(int j= j+1 ; j<n ; j++){
   if(arr[j] < arr[ ]){
        smallest = j ;}
   }
   swap(arr[i], arr[smallest]);
}
Void print(int array[ ], int size){
  for(int i=0 ; i<n ; i++)
   cout << array[i];
}
```

Date:

```
int main( ){
int size = 7;
int array[ ] = { 18, 12, 9, 20, 15, 10, 8 }
Selection (arr, size);
print (array, size);
return 0;
}
```

(3) array [35, 38, 32, 39, 31, 33, 36, 40]

size = 8

outer loop = n-1 = 7

innerloop → initialize from i+1 to (n) element → last

1st iter

i = 0 | j = i+1 = 0+1 = (1)

(35), 38, 32, 39, 31, 33, 36, 40

↓ assume as smin

(35), 38, 32, 39, 31, 33, 36, 40

(35), 38, 32, 39, 31, 33, 36, 40

35, 38, 32, 39, (31), 33, 36, 40

35, 38, 32, 39, 31, 33, 36, 40

35, 38, 32, 39, 31, 33, 36, 40

(35), 38, 32, 39, (31), 33, 36, 40

31, 38, 32, 39, 35, 33, 36, 40

2nd iter

i = 1 | j = i+1 = 1+1 = (2)

31, 38, 32, 39, 35, 33, 36, 40

Sorted | unsorted

31, (38), 32, 39, 35, 33, 36, 40

31, 38, 32, 39, 35, 33, 36, 40

31, 38, 32, 39, 35, 33, 36, 40

31, 38, 32, 39, 35, 33, 36, 40

31, (38), (32), 39, 35, 33, 36, 40

31, 38, 38, 39, 35, 33, 36, 40

Date:

5th iteration

i = 4

j = 4 + 1 = 5

8, 9, 10, 12, 15, 20, 18

sorted | unsorted

8, 8, 10, 12, 15, 20, 18

8, 9, 10, 12, 15, 20, 18

no swapping occur because the current min is not lesser than any remaining element.

6th iteration

i = 5

j = 5 + 1 = 6

8, 9, 10, 12, 15, 20, 18

sorted | unsorted

8, 9, 10, 12, 15, 20, 18

8, 9, 10, 12, 15, 18, 20

Final array.

Code:

```
#include <iostream>
using namespace std;
void selection(int arr[ ], int n){
  for(int i = 0 ; i<n-1 ; i++){
    int smallest = i ;}
for(int j = j+1 ; j<n ; j++){
    if(arr[j] < arr[ ]){
        smallest = j ;}
    }
    swap(arr[i], arr[smallest]);
}

Void print(int array[ ], int size){
  for(int i=0 ; i<n ; i++)
    cout << array[i];
}
```

Date:

```
int main( ){
int size = 7;
int array[ ] = { 18, 12, 9, 20, 15, 10, 8}
Selection (arr, size);
print (array, size);
return 0;
}
```

(3) array [35, 38, 32, 39, 31, 33, 36, 40]

size = 8

outer loop = n-1 = 7

innerloop → initialize from i+1 to (n) element → last

1st iter

i=0 & j = i+1 = 0+1 = (1)

35, 38, 32, 39, 31, 33, 36, 40

↓ assume as min

35, 38, 32, 39, 31, 33, 36, 40

35, 38, 32, 39, 31, 33, 36, 40

35, 38, 32, 39, 31, 33, 36, 40

35, 38, 32, 39, 31, 33, 36, 40

35, 38, 32, 39, 31, 33, 36, 40

35, 38, 32, 39, 31, 33, 36, 40

31, 38, 32, 39, 35, 33, 36, 40

2nd iter

i=1, j = i+1 = 1+1 = (2)

31, 38, 32, 39, 35, 33, 36, 40

Sorted | unsorted

31, 38, 32, 39, 35, 33, 36, 40

31, 38, 32, 39, 35, 33, 36, 40

31, 38, 32, 39, 35, 33, 36, 40

31, 38, 32, 39, 35, 33, 36, 40

31, 38, 32, 39, 35, 33, 36, 40

31, 38, 38, 39, 35, 33, 36, 40

Date:

3rd iter i=2 — i+1 = 2+1 = 3

31, 32, 38, 39, 35, 33, 36, 40

sorted | unsorted

31, 32, 38, 39, 35, 33, 36, 40

31, 32, 38, 39, 35, 33, 36, 40

31, 32, 38, 39, 35, 33, 36, 40

31, 32, 38, 39, 35, 33, 36, 40

31, 32, 33, 39, 35, 38, 36, 40

4th iter i=3 — i+1 = 3+1 = 4

31, 32, 33, 39, 35, 38, 36, 40

sorted | unsorted

31, 32, 33, 39, 35, 38, 36, 40

31, 32, 33, 39, 35, 38, 36, 40

31, 32, 33, 39, 35, 38, 36, 40

31, 32, 33, 35, 39, 38, 36, 40

5th iter i=4 — j = i+1 = 4+1 = 5

31, 32, 33, 35, 39, 38, 36, 40

sorted | unsorted

31, 32, 33, 35, 39, 38, 36, 40

31, 32, 33, 35, 39, 38, 36, 40

31, 32, 33, 35, 39, 38, 36, 40

31, 32, 33, 35, 36, 38, 39, 40

6th iter i=5 — j = i+1 = 5+1 = 6

31, 32, 33, 35, 36, 38, 39, 40

sorted | unsorted

31, 32, 33, 35, 36, 38, 39, 40

no swapping

7th iter i=6

31, 32, 33, 35, 36, 38, 39, 40

sorted | unsorted

31, 32, 33, 35, 36, 38, 39, 40

no swapping

Sorted array.

31, 32, 33, 35, 36, 38, 39, 40

Date: _____

4) array = [50, 46, 35, 28, 65, 76, 9, 18, 42]

size = 9

outer loop = n-1 = 9-1 = (8)

inner loop = initialize from (j+1) → end at (n)

1st iter
i=0 j = i+1 = 0+1 = (1)

50, 46, 35, 28, 65, 76, 9, 18, 42

assume as min

50, 46, 35, 28, 65, 76, 9, 18, 42

50, 46, 35, 28, 65, 76, 9, 18, 42

50, 46, 35, 28, 65, 76, 9, 18, 42

50, 46, 35, 28, 65, 76, 9, 18, 42

50, 46, 35, 28, 65, 76, 9, 18, 42 ...

9, 46, 35, 28, 65, 76, 50, 18, 42

2nd iter
i=1 j = i+1 = 1+1 = (2)

9, 46, 35, 28, 65, 76, 50, 18, 42

sorted | unsorted.

9, 46, 35, 28, 65, 76, 50, 18, 42

9, 46, 35, 28, 65, 76, 50, 18, 42

swap, swap

9, 46, 35, 28, 65, 76, 50, 18, 42 ...

9, 46, 35, 28, 65, 76, 50, 46, 42

3rd iter
i=2 j = i+1 = 2+1 = (3)

9, 18, 35, 28, 65, 76, 50, 46, 42

sorted | unsorted

9, 18, 35, 28, 65, 76, 50, 46, 42

9, 18, 28, 35, 65, 76, 50, 46, 42

4th iter
i=3 j = i+1 = 3+1 = (4)

9, 18, 28, 35, 65, 76, 50, 46, 42

sorted | unsorted

9, 18, 28, 35, 65, 76, 50, 46, 42

No swapping occurs because array is SI

5th iter
i=4 j = i+1 = 4+1 = 5

9, 18, 28, 35, 65, 76, 50, 46, 42

sorted | unsorted.

9, 18, 28, 35, 65, 76, 50, 46, 42

9, 18, 28, 35, 42, 76, 50, 46, 65

Date:

6th itey

i=5 j=6

9,18,28,35,42,76,50,46,65

sorted unsorted

9,18,28,35,42,76,50,46,65

9,18,28,35,42,76,50,46,65

9,18,28,35,42,46,50,76,65

7th itey

i=6 j=7

9,18,28,35,42,46,50,76,65

sorted unsor

9,18,28,35,42,46,50,76,

9,18,28,35,42,46,50,7

9,18,28,35,42,46,65,76,

8th itey

i=7 j=8

9,18,28,35,42,46,65,76,50

There is no swapping occur.

so 9,18,28,35,42,46,65,76,50 is sorted array.

(5) Array = [102, 108, 101, 109, 117, 103, 120, 138, 128, 120]

Size = 10

outer loop = size-1 = 10-1 = (9)

inner loop start from (i+1) → end at (n) less than

1st itey

i=0 j=i+1=0+1=(1)

102, 108, 101, 109, 117, 103, 120, 138, 128, 120

102, 108, 101, 109, 117, 103, 120, 138, 128, 120

comparing of remaining elements

102, 108, 101, 109, 117, 103, 120, 138, 128, 120

101, 108, 102, 109, 117, 103, 120, 138, 128, 120

Date:

2nd iteration
i = 1

$j = i+1 = 1+1 = 2$

101, 108, 102, 109, 117, 103, 120, 138, 128, 120
sorted | unsorted

101, 108, 102, 109, 117, 103, 120, 138, 128, 120

101, 102, 108, 109, 117, 103, 120, 138, 128, 120

3rd iter
i = 2

$j = 2+1 = 3$

101, 102, 108, 109, 117, 103, 120, 138, 128, 120
sorted | unsorted

101, 102, 108, 109, 117, 103, 120, 138, 128, 120

101, 102, 103, 109, 117, 108, 120, 138, 128, 120

4th iter
i = 3

$j = 3+1 = 4$

101, 102, 103, 109, 117, 108, 120, 138, 128, 120
sorted | unsorted

101, 102, 103, 109, 117, 108, 120, 138, 128, 120

101, 102, 103, 108, 117, 109, 120, 138, 128, 120

5th iter
i = 4

$j = 4+1 = 5$

101, 102, 103, 108, 117, 109, 120, 138, 128, 120
sorted | unsorted

101, 102, 103, 108, 117, 109, 120, 138, 128, 120

101, 102, 103, 108, 109, 117, 120, 138, 128, 120.

6th iter
i = 5

$j = 5 + 1 = 6$

101, 102, 183, 108, 109, 117, 120, 138, 120, 120
sorted — unsorted.

101, 102, 103, 108, 109, 117, 120, 138, 128, 120

There is no sorting occur because array[j] !< array[~~j+1~~] smallest

7th iter
i = 6

$j = 6 + 1 = 7$

101, 102, 103, 108, 109, 117, 120, 138, 128, 120
sorted — unsorted.

101, 102, 103, 108, 109, 117, 120, 138, 128, 120

101, 102, 103, 108, 109, 117, 120, 138, 128, 120
no swapping

8th iter
i = 7

$j = 7 + 1 = 8$

101, 102, 103, 108, 109, 117, 120, 138, 128, 120
sorted — unsorted

101, 102, 103, 108, 109, 117, 120, 138, 128, 120

101, 102, 103, 108, 109, 117, 120, 128, 138

9th iter
i = 8

$j = 8 + 1 = 9$

101, 102, 103, 108, 109, 117, 120, 120, 128, 138
no swapping

Date:

Insertion Sorting

[5, 2, 8, 1, 9]
0 1 2 3 4

outer loop i=1 = [2]

inner loop j=0 = [5]

current element / key = arr[i] = [2] ✓ compare

[5] > [2]

shift array to right

array [1] = array [0] = 5

j-- = -1

outer loop → (i=1) → (<n)

j<0 inner loop exit

Insert key [2] at arr [0]

Array [11, 1, 5, 4, 3, 2]

outer loop = (i start) → (<n)

Index 0 1 2 3 4 5

11, 1, 5, 4, 3, 2 i=1, j=0

11, 1, 5, 4, 3, 2

1, 11, 5, 4, 3, 2 i=2, j=1

1, 5, 11, 4, 3, 2 i=3, j=2

1, 4, 5, 11, 3, 2 i=4, j=3

1, 3, 4, 5, 11, 2 i=5, j=4

1, 2, 3, 4, 5, 11

sorted array.

Explanation:

Iteration 1

Date:

Index 0 1 2 3 4 5

[11, 1, 5, 4, 3, 2]

i = 1
j = 0

key / current element = arr[1] = 1

Compare arr[0] with arr[1] (current element)

∴ 11 > 1 shift arr[0] to right

arr[1] = arr[0] = 11

j-- ⇒ -1

j is less than 0 so inner loop exit

insert key (1) → to arr[0]

Iteration 2:

[1, 11, 5, 4, 3, 2]

i = 2
j = 1

current element = arr[2] = 5

compare array[1] with current elem

∴ 11 > 5 shift (11) to the right

j--

insert key (5) to arr[1]

arr[0] < 5 No swapping

j-- ⇒ -1

j is less than 0 so inner loop end

Iteration 3:

[1, 5, 11, 4, 3, 2]

i = 3
j = 2

key → arr[3] = 4

compare arr[2] with key

∴ 11 > 4 → shift array[3] / key to right

arr[3] = arr[2] = 11

movement of 4 to the arr[2] (swapping occur)

j--

key = 4

compare 4 with 5

∴ 5 > 4 → shift 5 to

Date:

[1,4,5,11,3,2]

array[1] = ary[2] = 5

j-- No swapping occurs.

Iteration 4:

[1,4,5,11,3,2]

i=4

j=3

key = [4] = 3

compare 3 with array[3] that is "11"

∴ 11 > 3 so 11 move right to 3

array[4] = ary[3] = 11

insert key to array[3] position.

j--

j=2

arry[2] > arr[3]

5 > 3 → move 5 to the right position

arry[3] = arr[3] = 5

insert 3 to arr[3] ✓ [1,4,3,5,11,2]

j--

j=1

array[1] > array[2]

4 > 3

move 4 to right position.

array[1] = arr[2] = 4

insert 3 tor [arr1] ——— [1,3,4,5,11,2]

j=0 ——— j-- ——— No swapping

j-- j<0 inner loop end.

Iteration 5:

[1,3,4,5,11,2]

i=5

key = 2 j=4

compare 2 with 11 then 5, 4, 3. That are greater than 2 so swapping occur and final array will be

[1,2,3,4,5,11,2]

Date:

(2) [18, 12, 9, 20, 15, 10, 8]

initially outer loop start from array index [1] but inner loop start from '0' index

then i loop increases but with increase of j loop compare the back element of it if greater than its element that insert the element to it and swapping occur.

i=1 j=0	18, 12, 9, 20, 15, 10, 8
i=2 j=1	12, 18, 9, 20, 15, 10, 8
i=3 j=2	9, 12, 18, 20, 15, 10, 8 — No swapping occur.
i=4 j=3	9, 12, 18, 20, 15, 10, 8
i=5 j=4	9, 12, 15, 18, 20, 10, 8
i=6 j=5	9, 10, 12, 15, 18, 20, 8
	8, 9, 10, 12, 15, 18, 20 — array after sorting

Iteration in pseudocode

(1) 1st element is considered sorted.
Compare 12 (2nd element) to left element
12 < 18 shift 12 to left

(2) Compare 9 (3rd element) to compare remaining elements to its left

(3) 9 < 18 and also < 12.
Then shift 9 to left side of 12

(4) Compare 20 with oth element no one is less so no swapping occur.

(5) Compare 15 (5th element) with other element
15 < 20 and 18 so swapping occur

(6) compare remaining elements one

Date:

3) array = [35, 38, 32, 39, 31, 33, 36, 40]

If indi[i] > index[j] then swapping occur.

i=1 j=0	35, 38, 32, 39, 31, 33, 36, 40	No swapping occur.
i=2 j=1	35, 38, 32, 39, 31, 33, 36, 40	
i=3 j=2	32, 35, 38, 39, 31, 33, 36, 40	No swapping occur.
i=4 j=3	32, 35, 38, 39, 31, 33, 36, 40	
i=5 j=4	31, 32, 35, 38, 39, 33, 36, 40	
i=6 j=5	31, 32, 33, 35, 38, 39, 36, 40	
i=7 j=6	31, 32, 33, 35, 36, 38, 39, 40	No swapping occur.

Iteration in pseudocode:

1st element is considered as sorted
compare 2nd element with index [0] there is no swapping occur because element at (1) index is greater than [0].

compare 3rd element (32) with remaing elements of left:
32 > 38 and 35 so 32 insert left to the 35

compare(39) 4th element with remaing elemet of its left side then there is no any less element to its then no swapping occur.

compare 5th element (31) to the remaing of its left sides
31 > (39, 38, 35, 32) so the so 31 insert to the left of 32

compare 6th element (33) with remaing elents ato its remaing elemt so its is smaller then 39, 38, 35 so the swapping occur.

compare 7th elemet (36) with other elements to its left
36 < 38 and 39 so swapping occur.

compare 8th elemt that is greater then all elm
So no swapping occur.

Date:

(4) [50, 40, 35, 28, 65, 76, 9, 18, 42]

if arr[i] > arr[j] then swapping occur.

i=1, j=0	50, 40, 35, 28, 65, 76, 9, 18, 42
i=2, j=1	40, 50, 35, 28, 65, 76, 9, 18, 42
i=3, j=2	35, 40, 50, 28, 65, 76, 9, 18, 42
i=4, j=3	28, 35, 40, 50, 65, 76, 9, 18, 42 No
i=5, j=4	28, 35, 40, 50, 65, 76, 9, 18, 42 No swapping occur.
i=6, j=5	28, 35, 40, 50, 65, 76, 9, 18, 42
i=7, j=6	9, 28, 35, 40, 50, 65, 76, 18, 42
i=8, j=7	9, 18, 28, 35, 40, 50, 65, 76, 42
	9, 18, 28, 35, 40, 50, 65, 76, 42 sorted array

Iteration in pseudocode:

- 1st element consider to be sorted.
- compare (2)nd element (40) with 1st elmt (50) 40 < 50 so swapping occur and 40 move
- left to the 1st element.
- compare 3rd elemnt (35) with remaing of its left sided Then 35 is < 40, so then it insert left to 40
- compare 4th element (28) with remaing elmts of inside that is smaller the 50, 40, 35 so swapping occur.
- compare 5th element (65) to its left remaining no swaping occur compare 6th element then is also no swapping occur.
- compare 7th, 8th, 9th elmnt in this way sorted array recieved.

9, 18, 28, 35, 40, 40, 65, 76, 42

Date:

5) Array = [102, 108, 101, 109, 117, 103, 120, 138, 128, 120]

Size = 10

swapping کے لیے arr[j] سے بڑا arr[i] کرنا

initially arr[i] > arr[j]

start from i=1 to < n

start from j=0.

i=1, j=0	102, 108, 101, 109, 117, 103, 120, 138, 128, 120 (NO swapping occur)
i=2, j=1	102, 108, 101, 109, 117, 103, 120, 138, 128, 120
i=3, j=2	101, 102, 108, 109, 117, 103, 120, 138, 128, 120 (NO swapping occur)
i=4, j=3	101, 102, 108, 109, 117, 103, 120, 138, 128, 120 (NO swapping)
i=5, j=4	101, 102, 108, 109, 117, 103, 120, 138, 128, 120
i=6, j=5	101, 102, 103, 108, 109, 117, 120, 138, 128, 120 (NO swapping)
i=7, j=6	101, 102, 103, 108, 109, 117, 120, 138, 128, 120 (NO swapping)
i=8, j=7	101, 102, 103, 108, 109, 117, 120, 138, 128, 120
i=9, j=8	101, 102, 103, 108, 109, 117, 120, 128, 138, 120
	101, 102, 103, 108, 109, 117, 120, 120, 128, 138 (Sorted array.)

Iteration and swapping in pseudocode:

In Insertion sorting i that is outer loop always start with [1] index that tell us about the iterations then j loop start from [0] index and with i,j index i^{th} element compare with j^{th} element ~~i element are less the j element then~~ swapping occur and insert the less element to the left side of j^{th} (greater) elements.

Date:

Code:

```
#include <iostream>
using namespace std;
void insertion_sort (int arr[], int size){
for (int i=1; i<n; i++){
   int key = arr[i];
int   j = i-1;
while (j>=0 && arr[j] > key) {
   arr[j+1] = arr[j];
   j--;
}
   arr[j+1] = key;
   }
}
   int main ( )
int arr [] = {102,108,101,109,117,103,120,138,128,120};
   int size = 10;
for (int i=0 ; i<size ; i++){
   cout << arrey [i];
}
insertion_sort (arr, size);
for(int i=0 ; i<size ; i++){
   cout << arr[i];
}
   return 0;
}
```

(unsorted array)

(sorted array)

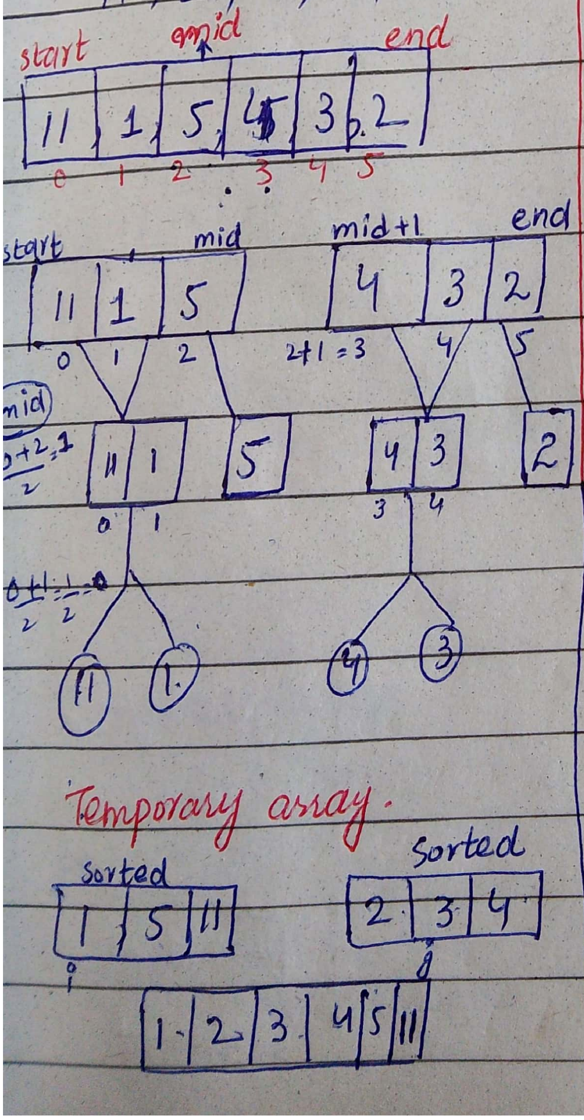
Date:

Merged array

Split the array into two halves
Sort each half seperately
Merge the two sorted halves into one.

اگر Array Numbers ~~even~~ ہیں تو 2 حصوں میں divide ہو جائے گا جیسا کہ right and left half ان دونوں کو seperately sort کیا جائے گا اور پھر ان دونوں کو merge کیا جائے گا compare کر کے سب سے چھوٹا element پہلے رکھا جائے گا اور last میں بڑا element

① [11, 1, 5, 4, 3, 2]



The elments in that's array are even so we divide into Two halves. and by finding the mid point using formula.

$$mid = start + \frac{(end - start)}{2}$$

$$= \frac{0 + (5-0)}{2} = \frac{5}{2} = 2$$

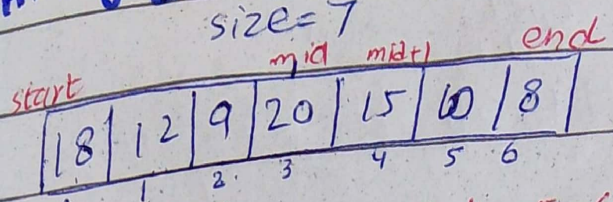
The divide the array into two parts. Sort them by spliting all the elements of array and find the mid point of that array and convert the whole array into single elements

At the end sorted left and right sides merges. Then it is copied to the orignal array.

Date:

Array [18, 12, 9, 20, 15, 10, 8]

size = 7

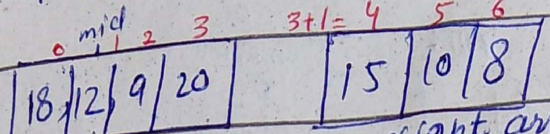


Find end by start of left arr and end of left arr

$\frac{0+3}{2} = \frac{3}{2} = 1$

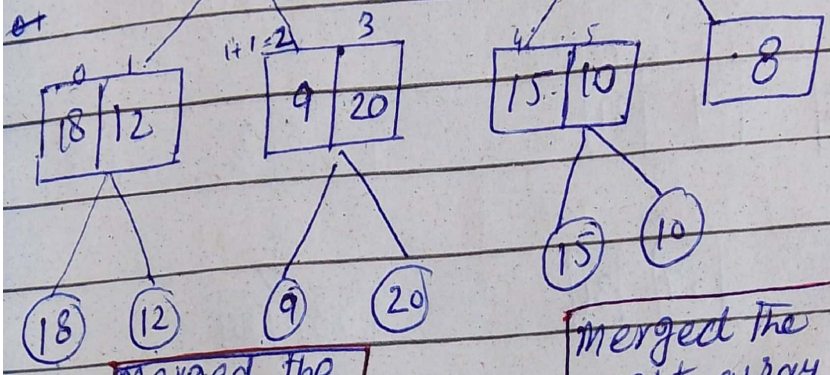
Index = 0

mid $\frac{0+6}{2} = \frac{6}{2} = 3$



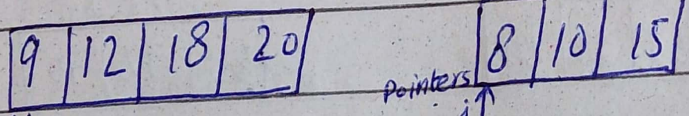
left array

right array



Merged the left array

Merged the right array



It is stored in temporary array.

Then pointers move on this array and merged them by using condition

arr[i] < arr[j]

ar[i] → temp array

i++

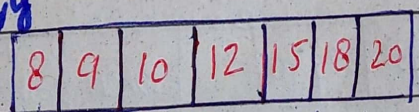
else

arr[j] < arr[i]

arr[j] → temp arr

j++

Temporary array



merged array.

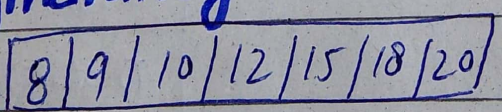
```
For (indx = 0 ; indx < temporary size ; indx++)
Arr[indx + start] = temporary[indx]
```

indx + start

↓

To store temp array in original

Original array

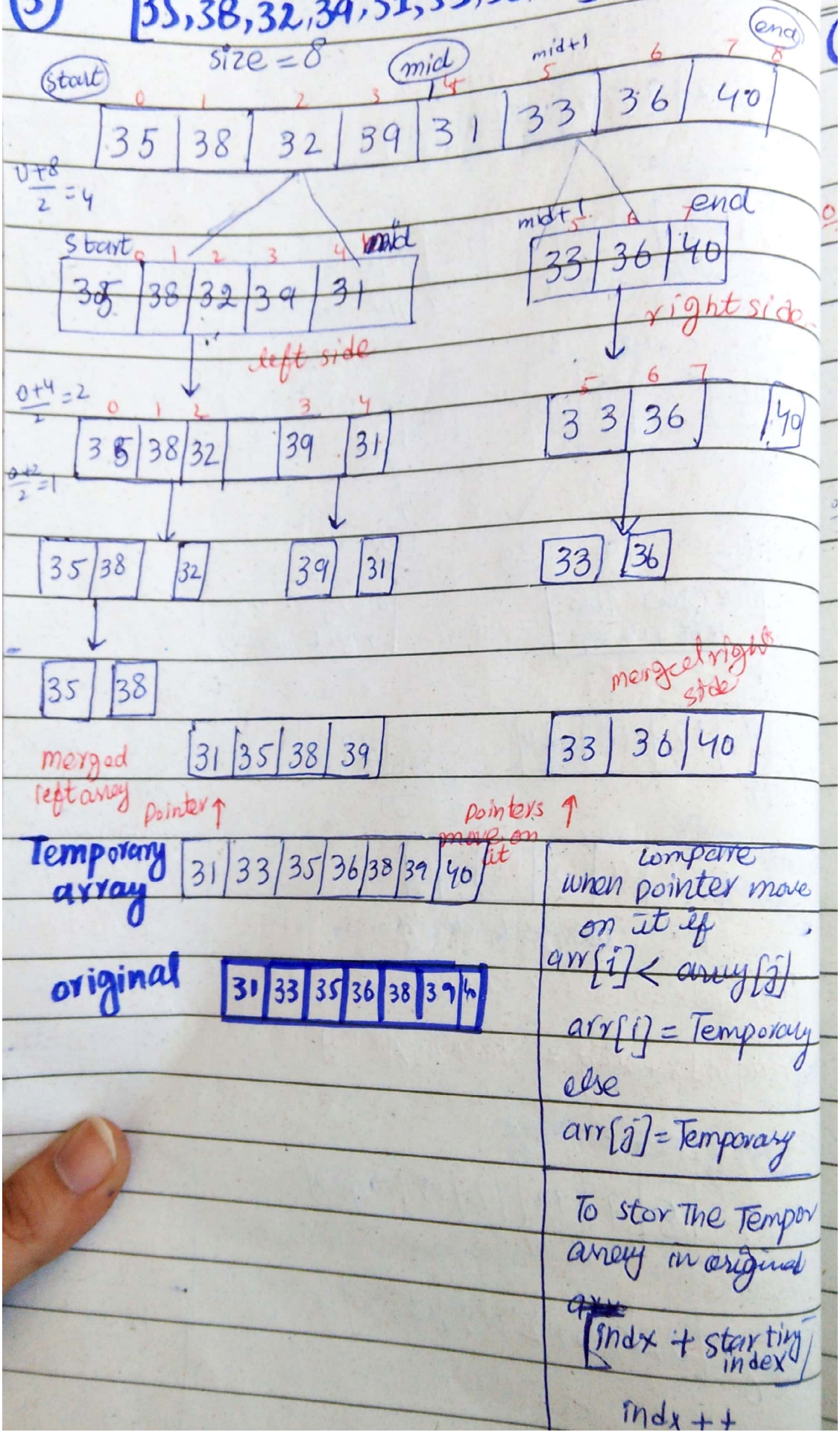


Final array = [8, 9, 10, 12, 15, 18, 20]

Date:

(3) [35, 38, 32, 39, 31, 33, 36, 40]

size = 8



compare when pointer move on it if

arr[i] < arr[j]

arr[i] = Temporary

else

arr[j] = Temporary

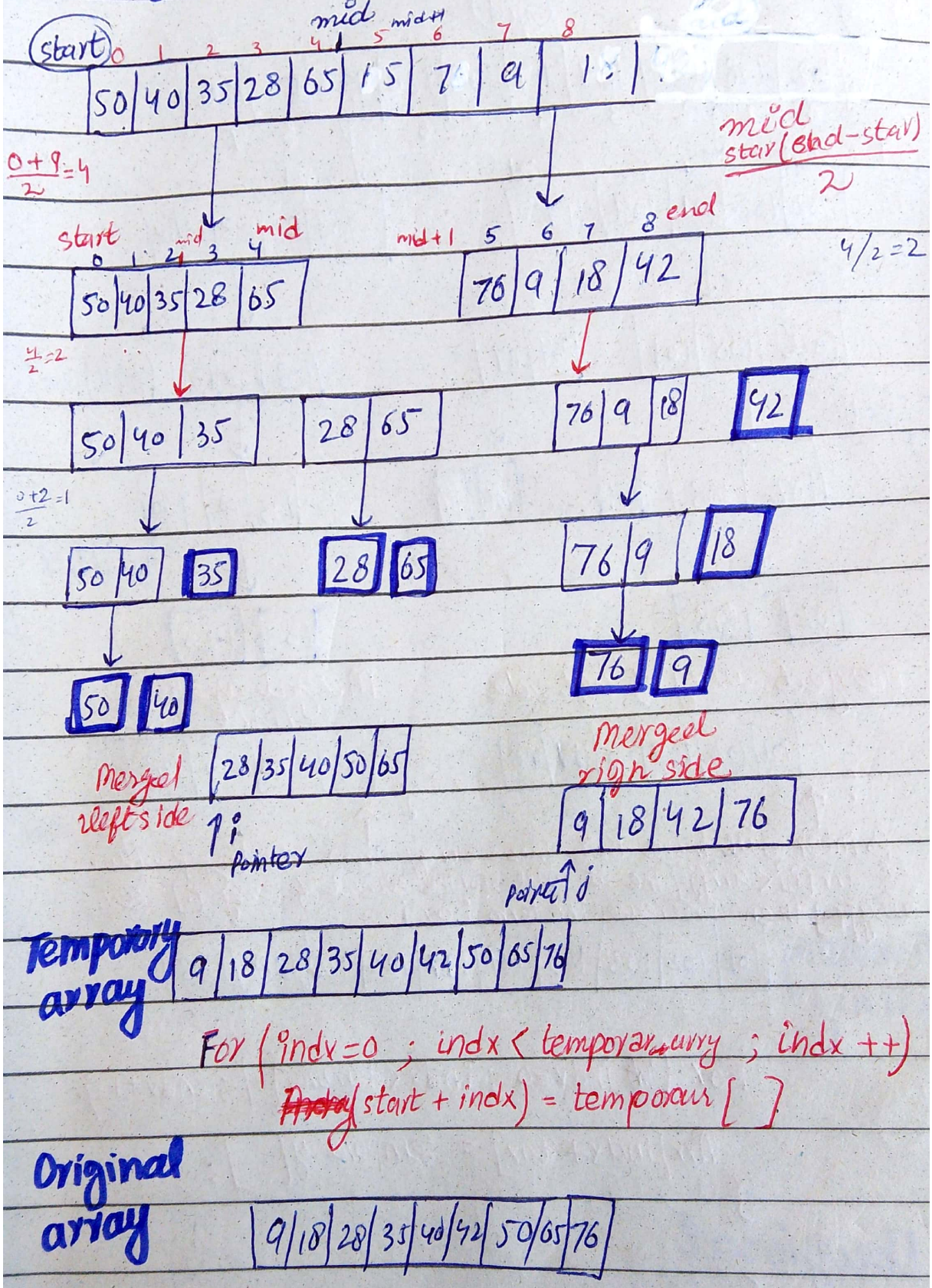
To stor the Tempor array in original

[indx + starting index]

indx ++

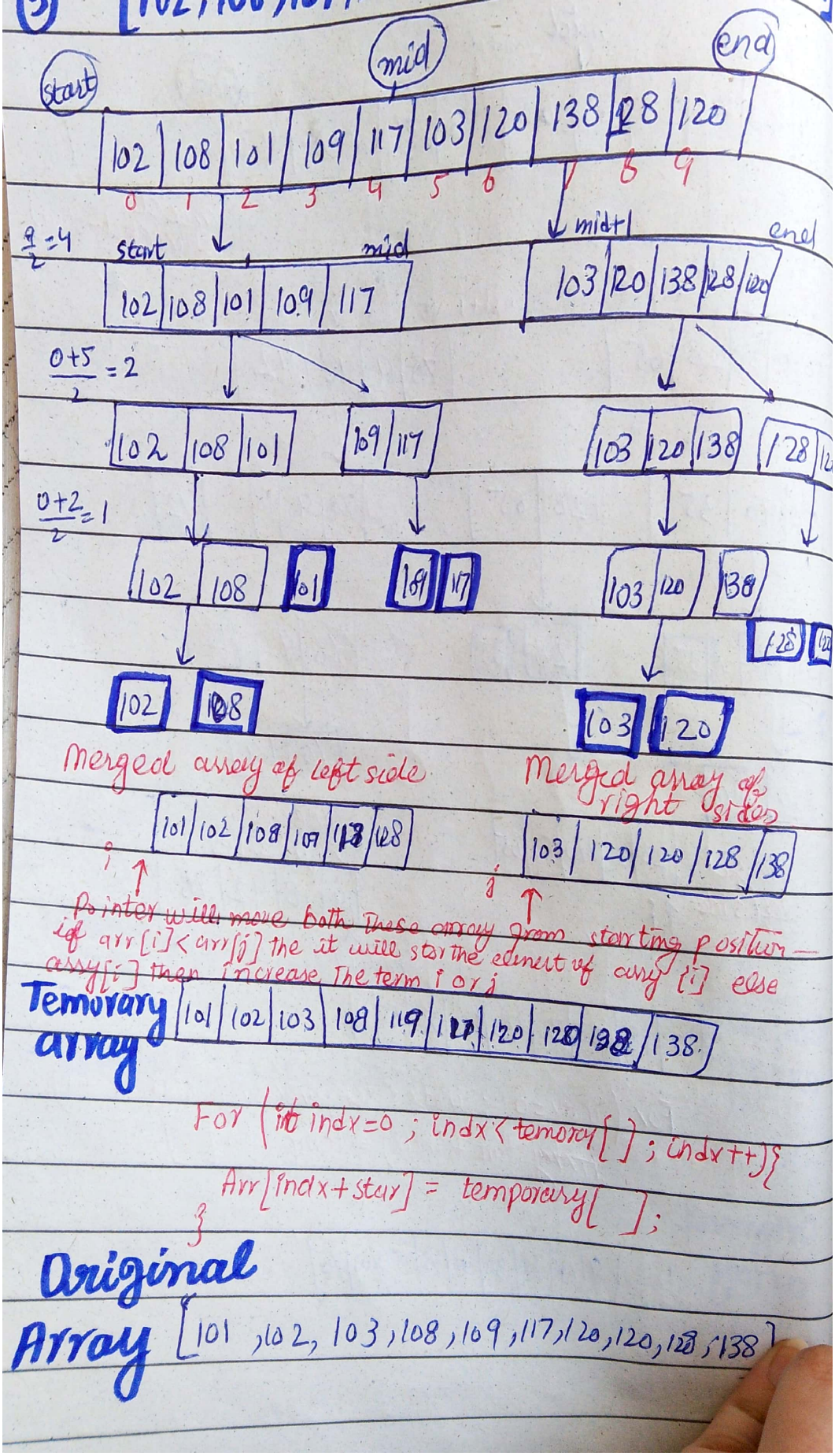
Date: _____

(4) [50, 40, 35, 28, 65, 76, 9, 18, 42]



Date:

(5) [102, 108, 101, 109, 117, 103, 120, 138, 128, 120]



Date: _____

Quick Array

chose pivet : pick an element in the list (usually the last element).

Rearrange the list - Move all the element smallest elemt left to Pivot and large element to the right side of pivot.

① Array [11, 1, 5, 4, 3, 2]

11, 1, [5], 4, 3, 2

1, 2, 3, 4, 5, 11

② Array [18, 12, 9, 20, 15, 10, 8]

18, [12], 9, 20, 15, 10, 8

8, 9, 10, 12, 15, 18, 20

③ Array [35, 38, 32, 39, 31, 33, 36, 40]

[35], 38, 32, 39, 31, 33, 36, 40

31, 32, 33, 35, 36, 38, 39, 40

④ Array [50, 46, 35, 28, 65, 76, 9, 18, 42]

50, 46, 35, [28], 65, 76, 9, 18, 42

9, 18, 28, 35, 42, 46, 50, 65, 76

Explanation

Quicksort is one of the efficient method of sorting algorithm that is based on swap and conquer approach

Date: _____

From which one pivot point from the whole array is selected and right side element push right to pivot and smaller element push left to the pivot point.

---

5) [102, 108, 101, 109, 117, 103, 120, 138, 128, 120]

[02, 108, 101, 109, (117) 103, 120, 138, 128, 120
selected as pivot point

101, 102, 103, 108, 109 (117) 120, 120, 128, 138
that is sorted array.

Date: _____

Radix Sorting

Array [11, 1, 5, 4, 3, 2]

11	1[1]	01
01	0[1]	02
05	0[2]	03
04	0[3]	04
03	0[4]	05
02	0[5]	11

Sorted array

{01, 02, 03, 04, 05, 11}.

Explanation:

In every sorting we observe that we compare the elements with one another. But here we can't compare the elements It is no comparative base catagory. First we can equal strength in this sorting. First element has 2 digits so every element will be in 3 digit We add zero then righ sided elements firstly sorted then right side elements sorted in ascending order as finally the sorted array is recieved.

Date:

Array [18, 12, 9, 20, 15, 10, 8]

18, ~~12~~	20	08
12	10	09
09	12	10
20	15	12
15	18	15
10	08	18
08	09	20

Sorted array = {08, 09, 10, 12, 15, 18, 20}

Array [35, 38, 32, 39, 31, 33, 36, 40]

35	40	31
38	31	32
32	32	33
39	33	34
31	35	35
33	36	36
36	38	38
40	39	39

Sorted array {31, 32, 33, 34, 35, 36, 39, 40}.

Array [50, 46, 35, 28, 65, 76, 09, 18, 42]

50	50	09
46	42	18
35	35	28
28	65	35
65	76	42
76	46	46
09	28	~~65~~
18	18	65
42	09	76

Sorted array = {09, 18, 28, 35, 42, 46, 50, 65, 76}

Date: _____

Array [102, 108, 101, 109, 117, 103, 120, 138, 128, 120]

102	120	101	101
108	101	102	102
101	102	103	103
109	103	107	107
117	107	108	108
103	108	109	109
120	138	120	120
138	128	120	120
128	109	128	128
120	120	138	138

Sorted array = {101, 102, 103, 107, 108, 109, 120, 120, 128, 138}.