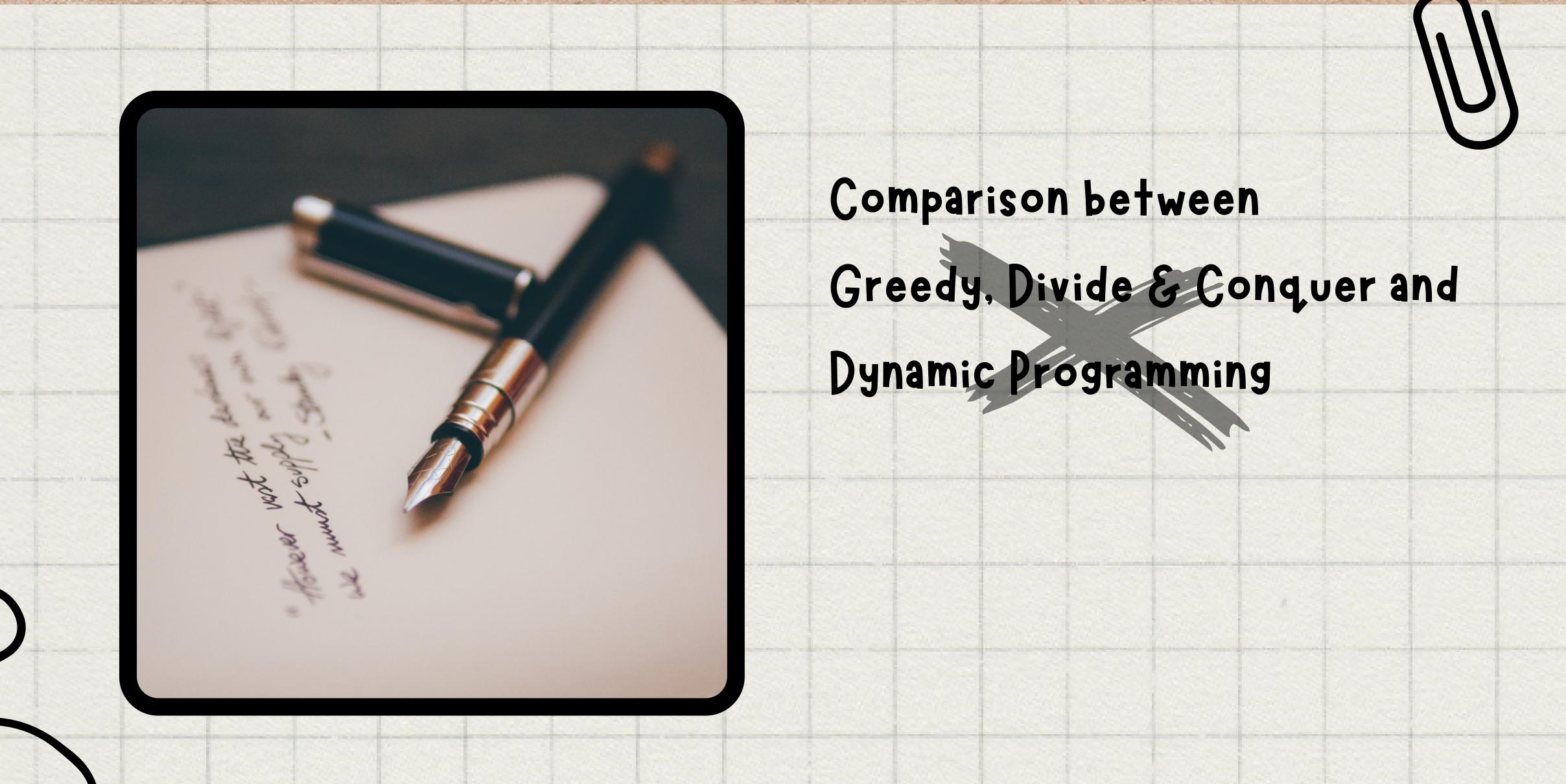


Week 9 - ADA

PROJECT MILESTONE

Jocelin | Nicholas | Tiffany

PREVIOUS TOPIC





WEIRD SORTING ALGORITHMS



Spaghetti

Identifying and then removing the longest spaghetti rod



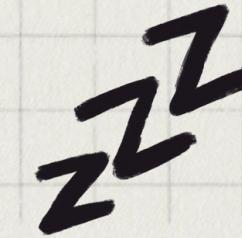
Stalin

Eliminating numbers that are not positioned correctly.



Sleep

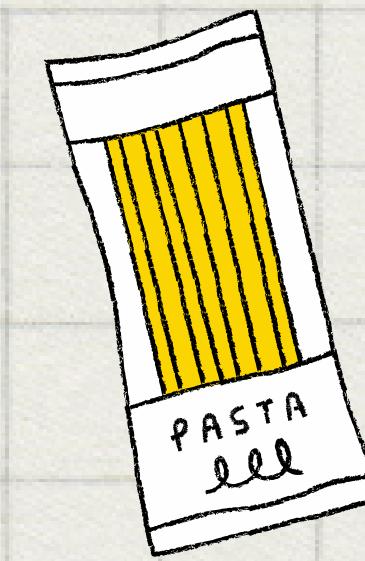
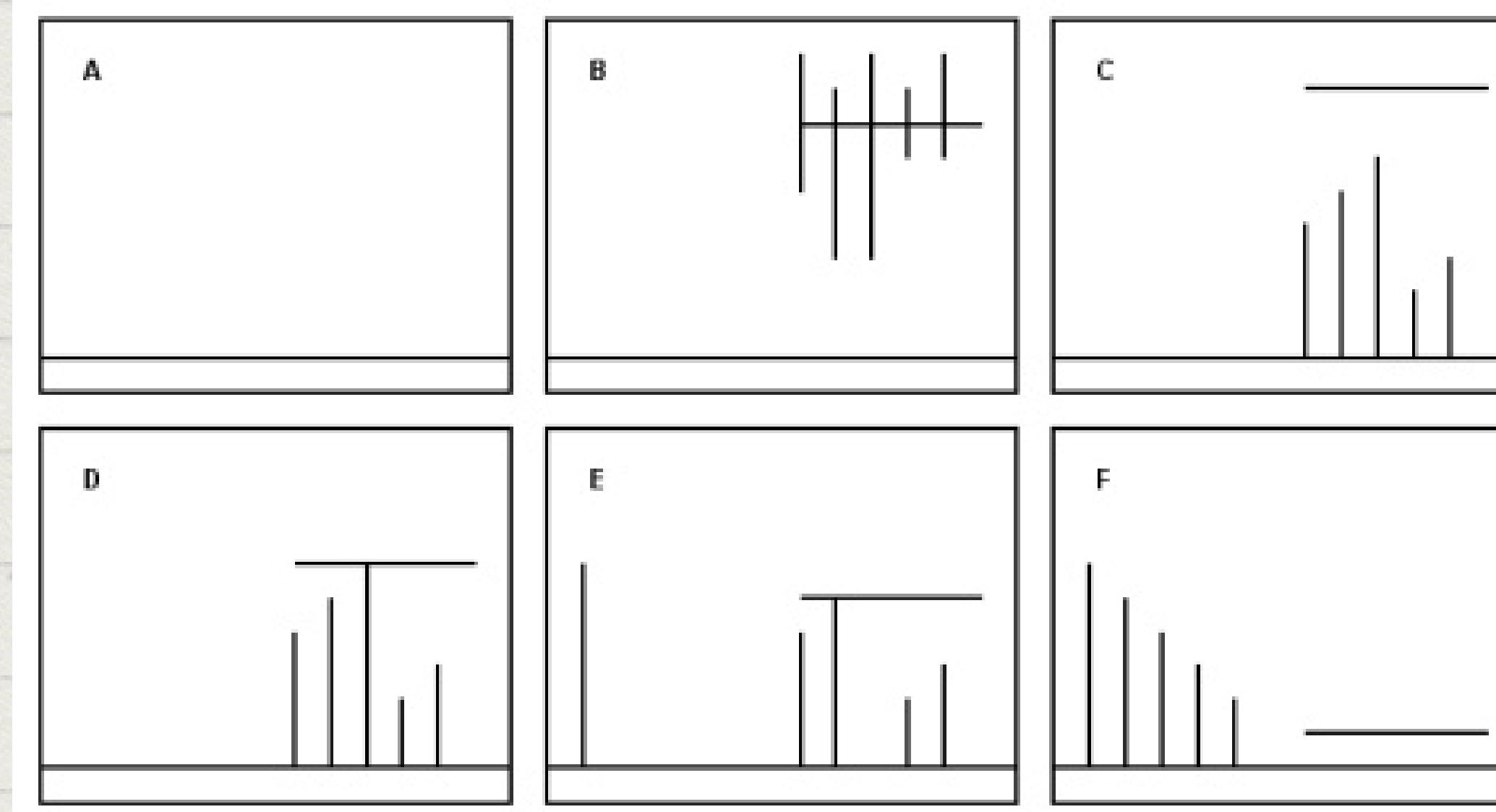
The element having the least amount of sleeping time wakes up first and the number gets printed



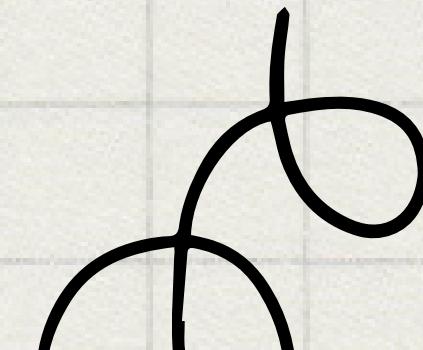
Cocktail Shaker

A variation of bubble sort. It traverses through a given array in both directions.





SPAGHETTI VISUALIZATION



FOR N IN ARRAY
CREATE A NEW THREAD
MAKE IT SLEEP FOR ARRAY[N]
MILLISECONDS
PRINT ARRAY[N]
END
WAIT FOR ALL PROCESSES TO
FINISH

SLEEP SORT PSEUDO-CODE





```
FUNCTION STALINSORT(A : LIST OF SORTABLE ITEMS)
    N := LENGTH(A)
    BIGGER := 0
    B SET EMPTY LIST

    FOR I := 0 TO N NOT INCLUSIVE DO
        IF A[I] >= BIGGER THEN
            BIGGER := A[I]
            B.PUSH(A[I])
        END IF
    END FOR

    RETURN B
END FUNCTION
```



STALIN SORT PSEUDO-CODE

Algorithm 1: Cocktail sort

```
Data: Input array A[]  
Result: Sorted A[]  
int i,swapped;  
do  
    swapped=0;  
    foreach i in 0 to length(A)-2 do  
        if A[i] > A[i + 1] then  
            swap(A[i],A[i+1]);  
            swapped=1;  
        end  
    end  
    if !swapped then  
        break do-while loop;  
    end  
    swapped=0;  
    foreach i in length(A)-2 to 0 do  
        if A[i] > A[i + 1] then  
            swap(A[i],A[i+1]);  
            swapped=1;  
        end  
    end  
while swapped;
```

COCKTAIL SHAKER PSEUDO-CODE





CODE DEMO TIME!



SLEEP SORT

```
#include <iostream>
#include <vector>
#include <thread>
#include <chrono>
#include <algorithm>
using namespace std;

template< typename It >
void sleepSort( const It& begin, const It& end )
{
    if ( begin == end )
        return;
    const auto [min, max] = minmax_element( begin, end );
    for ( auto it = begin; it != end; ++it )
    {
        thread( [min = *min, i = *it]
        {
            this_thread::sleep_for(chrono::milliseconds( i - min ) );
            cout << i << endl;
        } ).detach();
    }
    this_thread::sleep_for(chrono::milliseconds( *max - *min + 1 ) );
}

int main( int argc, char *argv[] )
{
    vector<int> v = { 20, 5, 15, 10, -5, };
    sleepSort(v.begin(), v.end() );

    return 0;
}
```

SORTED!

```
-5
5
10
15
20
```

STALIN SORT

```
1  #include <vector>
2  #include <iostream>
3  #define KEY StalinSort
4  #define minor main
5  #define pointer *
6  #define getPointer &
7  #define print std::cout
8  #define array std::vector
9  #define add push_back
10
11 //This implementation of stalin sort is easier to read and understand
12 template <ttypename T>
13 array<T> stalinSort(T pointer arr, size_t count)
14 {
15     array<T> orderedData;
16     orderedData.add(arr[0]);
17
18     unsigned int last = 0;
19     for(unsigned int i = 1;i < count;i++)
20     {
21         if (arr[i] >= orderedData[last])
22         {
23             last++;
24             orderedData.add(arr[i]);
25         }
26     }
27     return orderedData ;
28 }
29
30 int minor()
31 {
32     array<int> KEY ;
33     KEY.add(1) ;
34     KEY.add(3) ;
35     KEY.add(2) ;
36     KEY.add(5) ;
37     KEY.add(3) ;
38     KEY.add(8) ;
39     KEY.add(7) ;
40     KEY.add(9) ;
41
42     array<int> ordered = stalinSort<int>(getPointer KEY[0], KEY.size());
43
44     for (unsigned int i = 0 ; i < ordered.size() ; i++)
45         print << ordered[i] << "\n";
46
47     return 0 ;
48 }
```

C:\ Microsoft Visual Studio Debug Console
Sorted array: 1 3 5 8 9

SORTED!



MERCIFUL DICTATOR

```
1 //include<iostream>
2 //include<vector>
3 using namespace std;
4
5 // Function to sort the array
6 void variationStalinsort(vector<int> arr)
7 {
8     int j = 0;
9
10    while (true)
11    {
12        int moved = 0;
13
14        for (int i = 0;
15             i < (arr.size() - 1 - j); i++)
16        {
17            if (arr[i] > arr[i + 1])
18            {
19                vector<int>::iterator index;
20                int temp;
21                index = arr.begin() + i + 1;
22                temp = arr[i + 1];
23                arr.erase(index);
24                arr.insert(arr.begin() + moved, temp);
25                moved++;
26            }
27        }
28
29        j++;
30
31        if (moved == 0)
32        {
33            break;
34        }
35
36        cout << "Sorted array: ";
37        for (int i = 0; i < arr.size(); i++)
38        {
39            cout << arr[i] << " ";
40        }
41
42
43 // Driver Code
44 int main()
45 {
46     vector<int> arr = { 1, 3, 2, 5, 3, 8, 7, 9 };
47
48     // Function call
49     variationStalinsort(arr);
50 }
```

SORTED!

```
Microsoft Visual Studio Debug Console
Sorted array: 1 2 3 3 5 7 8 9
```



COCKTAIL SORT

```
1 #include <iostream>
2
3 using namespace std;
4
5 void CocktailSort(int a[], int n)
6 {
7     bool swapped = true;
8     int start = 0;
9     int end = n - 1;
10
11    while (swapped) {
12        //set default swapped variable to false
13        swapped = false;
14
15        // loop from left to right same as
16        // for (int i = start; i < end; ++i) {
17        for (int i = start; i < end; ++i) {
18            if (a[i] > a[i + 1]) {
19                swap(a[i], a[i + 1]);
20                swapped = true;
21            }
22
23            // if nothing gets swapped then break the loop cz its basically sorted
24            if (!swapped)
25                break;
26            // Repeat with end point moved back by 1
27            swapped = false;
28
29            --end;
30
31            for (int i = end - 1; i >= start; --i) {
32                if (a[i] > a[i + 1]) {
33                    swap(a[i], a[i + 1]);
34                    swapped = true;
35                }
36
37                // increase the starting point, because
38                // the last stage would have moved the next
39                // smallest number to its rightful spot.
40                //+start;
41                ++start;
42            }
43        }
44    }
45
46    /* Prints the array */
47    void printArray(int a[], int n)
48    {
49        for (int i = 0; i < n; i++)
50            printf("%d ", a[i]);
51        printf("\n");
52    }
53
54    // vroom vroom
55    int main()
56    {
57        int a[] = { 1, 3, 2, 5, 3, 8, 7, 9 };
58        int n = sizeof(a) / sizeof(a[0]);
59        CocktailSort(a, n);
60        printf("Sorted array :\n");
61        printArray(a, n);
62        return 0;
63    }
```

Microsoft Visual Studio Debug Console
Sorted array : 1 2 3 3 5 7 8 9

SORTED!

OUR NEXT STEP!



- test out 8 different sized
nearly, randomly and
reverse sorted arrays.
- Measure their run times
- Compare and analyze
- Conclude findings!

THANK you