ALTERNATING DISKS

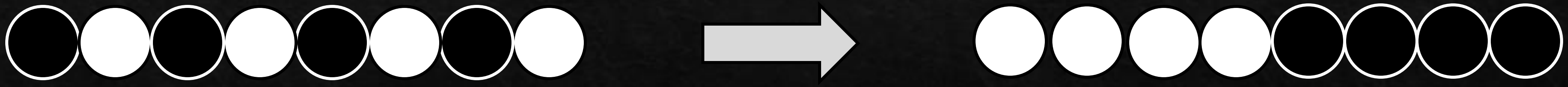
Course Coordinator

Prof.Gururaja H.S

INTRODUCTION

PROBLEM STATEMENT

ALTERNATING DISKS:
You have a row 2n disks of two colors , n dark and n light. They alternate : dark , light and so on. You want to get all the dark disks to the right end and all the lights disks to the left-hand end. The only moves you are allowed to make are those that interchange the positons of two neighbouring disks.



Design an algorithm for solving this puzzle and determine the number of moves it takes.

BRUTE FORCE TECHNIQUE

- Brute Force Algorithms refers to a programming style that does not include any shortcuts to improve performance.

- It instead relies on sheer computing power to try all possibilities until the solution to a problem is found.

- A brute-force algorithm is simple to implement , and will always find a solution if it exists.

- Its cost is proportional to the number of candidate solutions – which in many practical problems tends to grow very quickly as the size of the problem increases

Selection of design techniques:

LEFT TO RIGHT(BRUTE FORCE-BUBBLE SORT)

INPUT:A positive integer n and a list of 2n disks of alternating colors dark-light,starting with dark

OUTPUT:A list of 2n disks,the first n disks are light,the next n disks are dark and an integer m representing the number of moves to move the dark ones after the light ones

Here we will consider,
A)Light ball as –L
B)dark ball as –d
C)Starting ball will always be dark
D)Number of moves(swap)done during execution is stored in variable m

1. Scan from the leftmost
2. Compare with the immediate right one and do swap if necessary
3. When reaches the end, comes back to the leftmost
4. Repeat until it is sorted

```
 m = 0; //set m before entering in the loop
 SORTING I begins here---Right to left algorithm
 for (int j = 0; j < n; j++) {
 for (int i = 0; i < 2 * n; i++)
  {
  if (disk[i] == 'd' && disk[i+1] == 'l')
  {
 swap(disk[i], disk[i+1]); m++;
  }
 }
 }
```

DESIGN TECHNIQUE 2: LAWNMOVER

INPUT: a positive integer n and a list of 2n disks of alternating colors dark-light, starting with dark

OUTPUT: a list of 2n disks, the first n disks are light, the next n disks are dark, and an integer m representing the number of moves to move the dark ones after the light ones,

Here,we will consider
a)Light ball as - L
b)Dark ball as – d
c)Starting ball will always bs dark
d)Number of moves(swap) done during excecution is stored in variable m

1.Scan from the leftmost

2.Compare with the immediate right one and do swap if necessary.

3.When reaches end, start from the end moving backwards, compare with the immediate left one and do swap if necessary

4.repeat until it is sorted

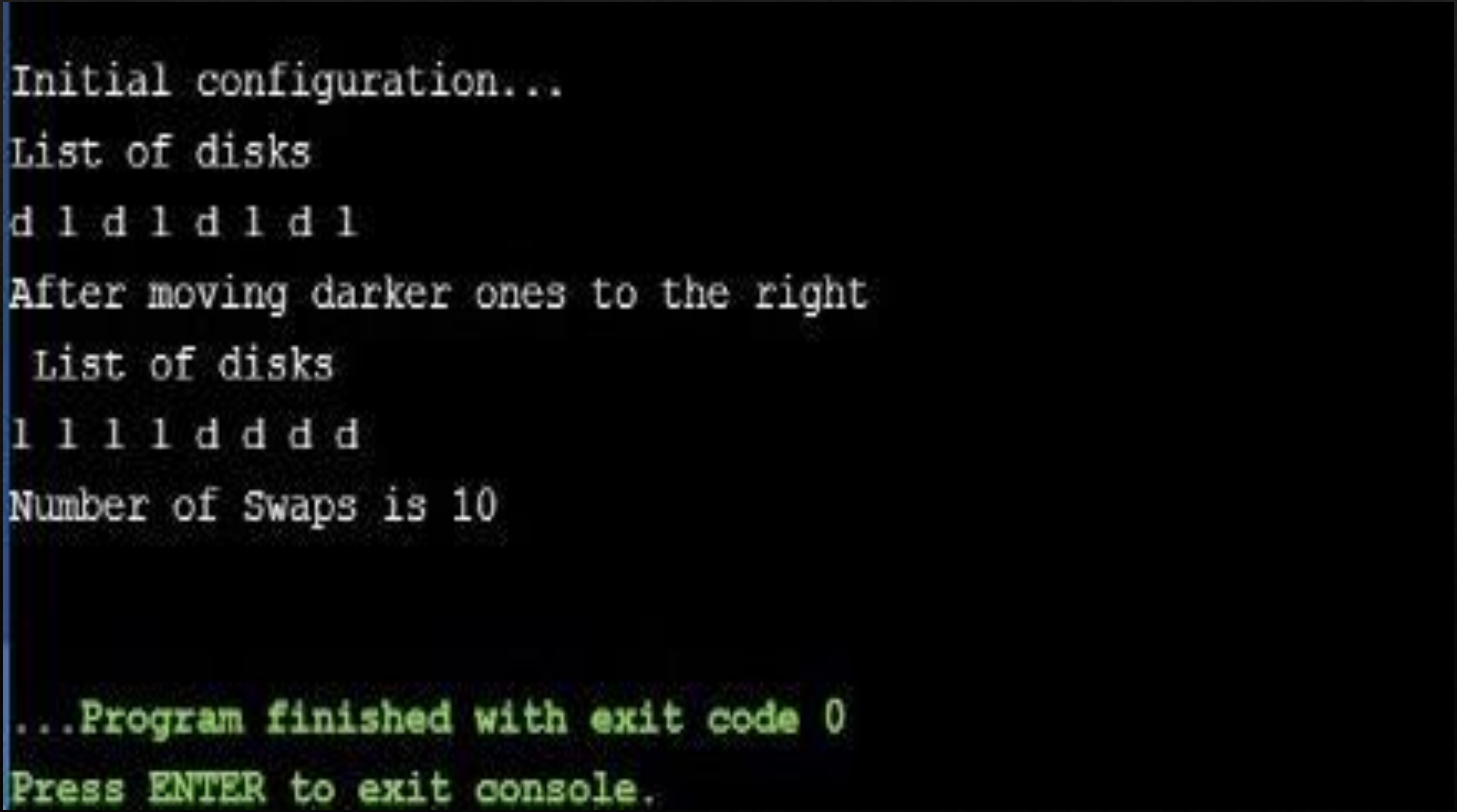
IMPLEMENTATION

LEFT TO RIGHT

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

void print(char* arr, int n) {
for (int i = 0; i < 2 * n; i++)

cout << arr[i] << " ";
cout << endl;
}

int main()
{
  int n, m = 0;  //n is the number of single colored, m is tof
swaps
```

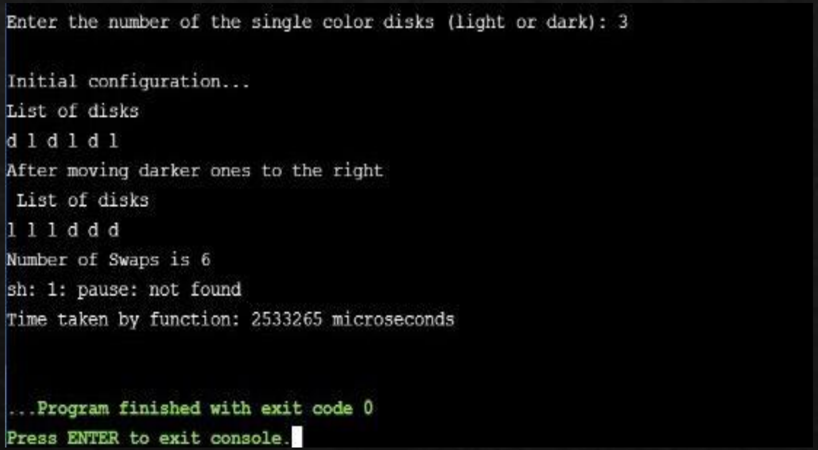
```
auto start = high_resolution_clock::now();
char disk[100];
//print out welcome message
cout << "technique# 2"  << "\nEnter the number of the single   color disks (light or dark): ";
cin >> n;
cout << endl;
cout << "Initial configuration...\nList of disks\n";
//intialization the alternating values
for (int i = 0; i < n; i++) {
disk[2 * i] = 'd';
disk[2 * i + 1] = 'l';
}
//print out the disk
print(disk, n);
for (int j = 0; j < n / 2 + 1; j++) {
for (int i = 0; i < 2 * n; i++) {
if (disk[i] == 'd' && disk[i + 1] == 'l') {
swap(disk[i], disk[i + 1]);
m++;
}
}
for (int i = 2 * n - 1; i >= 0; i--) {
if (disk[i] == 'l' && disk[i - 1] == 'd') {
swap(disk[i], disk[i - 1]);
m++;
}
}
}
cout << "After moving darker ones to the right\n List of disks\n";
//print out the sorted disk
print(disk, n);
auto stop = high_resolution_clock::now();
//print the number of swap
cout << "Number of Swaps is " << m << endl;
system("pause");
auto duration = duration_cast<microseconds>(stop - start);
cout << "Time taken by function: "
        << duration.count() << " microseconds" << endl;
}
```

OUTPUT

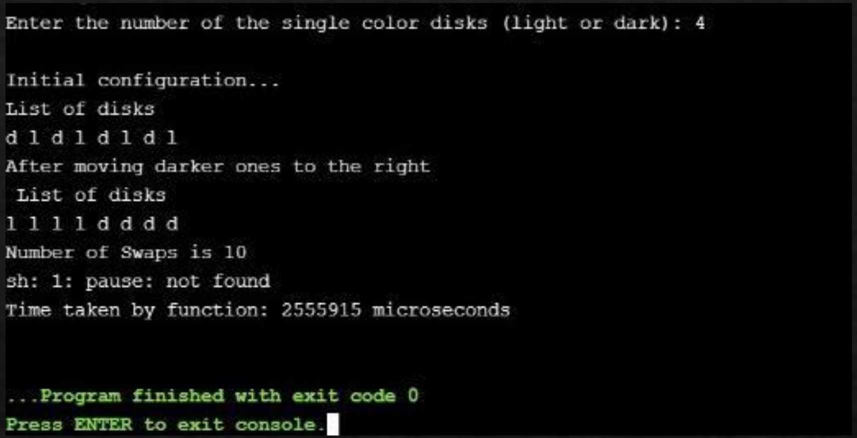


```
Initial configuration...
List of disks
d l d l d l d l
After moving darker ones to the right
 List of disks
l l l l d d d d
Number of Swaps is 10



...Program finished with exit code 0
Press ENTER to exit console.
```

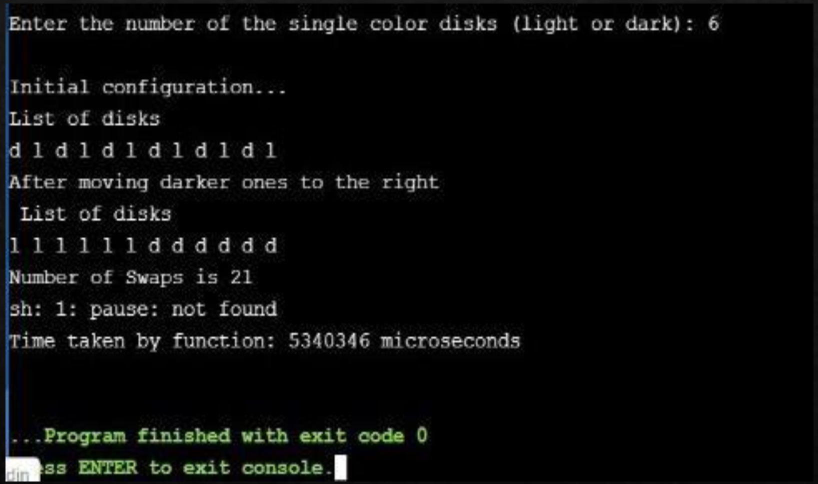
ANALYSING TIME EFFICIENCY



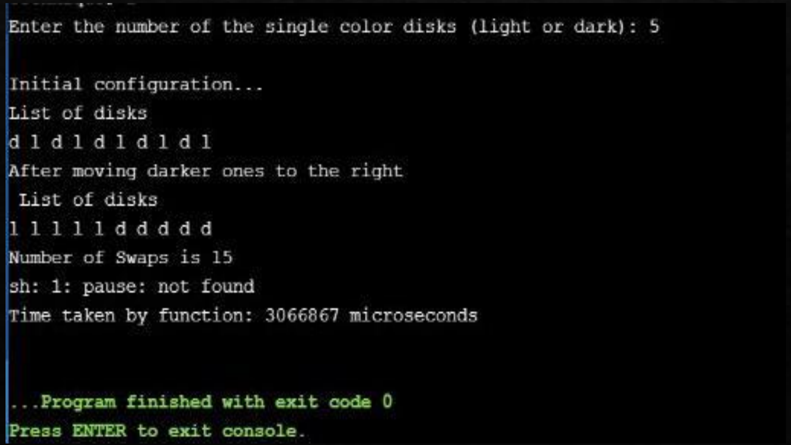
```
Enter the number of the single color disks (light or dark): 3

Initial configuration...
List of disks
d l d l d l
After moving darker ones to the right
 List of disks
l l l d d d
Number of Swaps is 6
sh: 1: pause: not found
Time taken by function: 2533265 microseconds


...Program finished with exit code 0
Press ENTER to exit console.
```



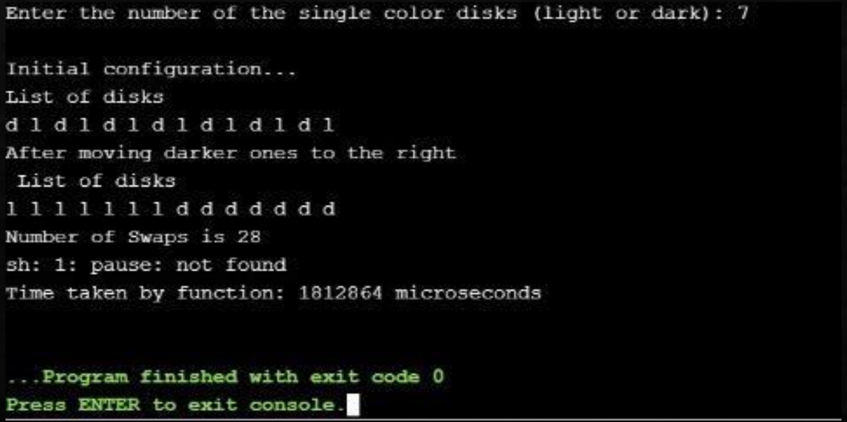
```
Enter the number of the single color disks (light or dark): 4

Initial configuration...
List of disks
d l d l d l d l
After moving darker ones to the right
 List of disks
l l l l d d d d
Number of Swaps is 10
sh: 1: pause: not found
Time taken by function: 2555915 microseconds


...Program finished with exit code 0
Press ENTER to exit console.
```



```
Enter the number of the single color disks (light or dark): 6

Initial configuration...
List of disks
d l d l d l d l d l d l
After moving darker ones to the right
 List of disks
l l l l l l d d d d d d
Number of Swaps is 21
sh: 1: pause: not found
Time taken by function: 5340346 microseconds


...Program finished with exit code 0
ss ENTER to exit console.
```



```
Enter the number of the single color disks (light or dark): 5

Initial configuration...
List of disks
d l d l d l d l d l
After moving darker ones to the right
 List of disks
l l l l l d d d d d
Number of Swaps is 15
sh: 1: pause: not found
Time taken by function: 3066867 microseconds


...Program finished with exit code 0
Press ENTER to exit console.
```



```
Enter the number of the single color disks (light or dark): 7

Initial configuration...
List of disks
d l d l d l d l d l d l d l
After moving darker ones to the right
 List of disks
l l l l l l l d d d d d d d
Number of Swaps is 28
sh: 1: pause: not found
Time taken by function: 1812864 microseconds


...Program finished with exit code 0
Press ENTER to exit console.
```

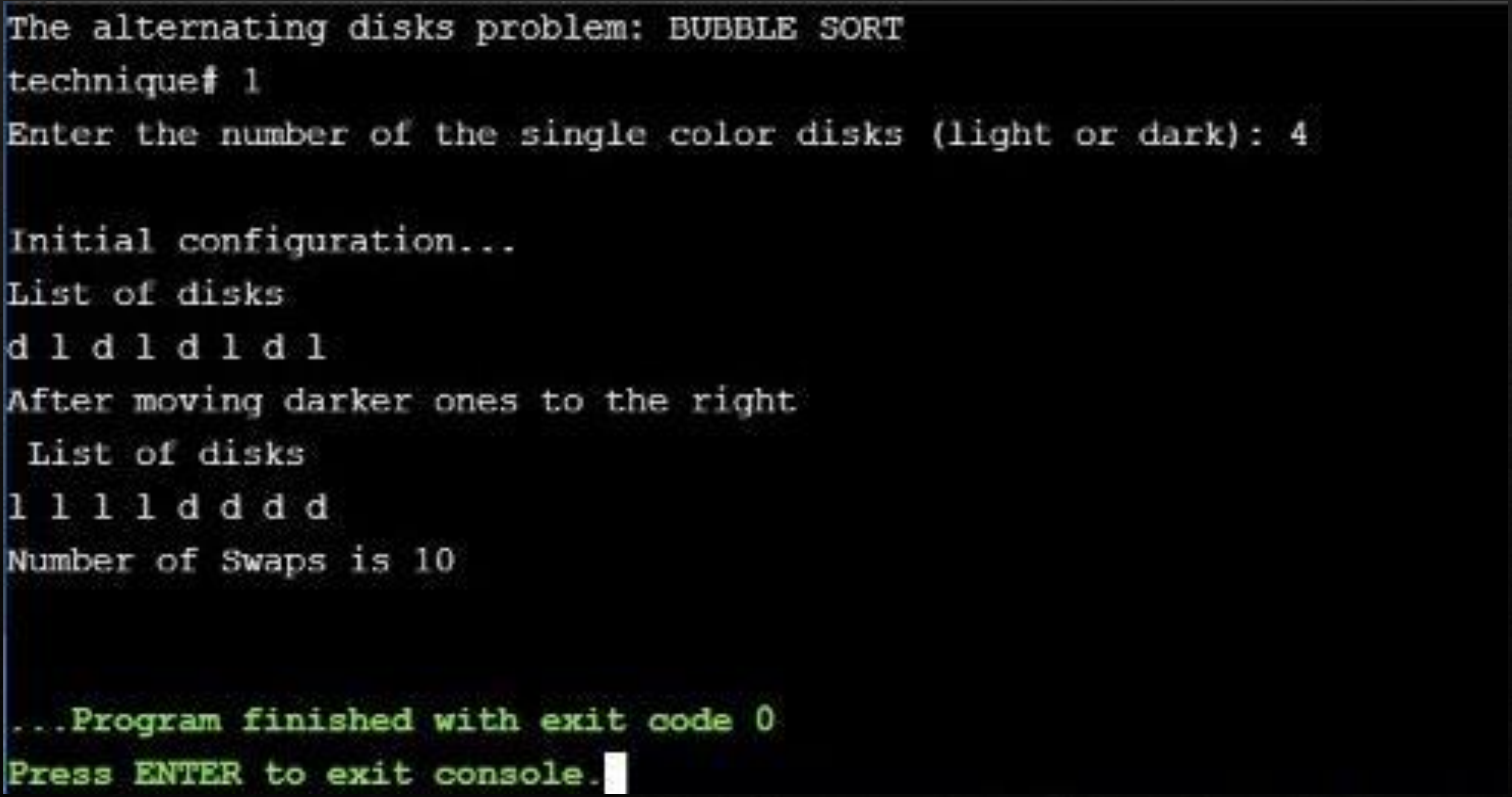
LAWNMOVER

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

void print(char* arr, int n) {
for (int i = 0; i < 2 * n; i++)
cout << arr[i] << " ";
cout << endl;
}
int main()
{
int n, m = 0;
//n is the number of single colored, m is the number of swaps
auto start = high_resolution_clock::now();

char disk[100];
//print out welcome message
cout << "The alternating disks problem: BUBBLE SORT\n";
cout << "technique# 1"  << "\nEnter the number of the single color disks (light or dark): ";
```

```
cin >> n;
cout << endl;
cout << "Initial configuration...\nList of disks\n";

//intialization the alternating values
for (int i = 0; i < n; i++) {
disk[2 * i] = 'd';
disk[2 * i + 1] = 'l';
}

//print out the disk

print(disk, n);

//SHUFFLING
for (int j = 0; j < n; j++) {

for (int i = 0; i < 2 * n; i++) {

if (disk[i] == 'd' && disk[i + 1] == 'l') {
swap(disk[i], disk[i + 1]);
m++;
}
}
}
```

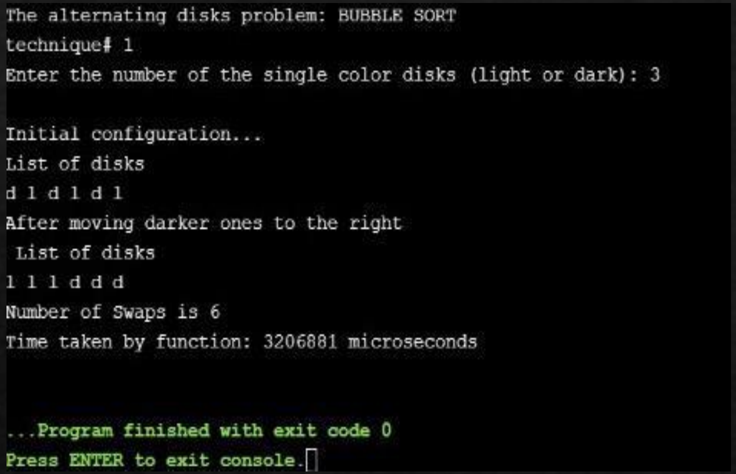
```
cout << "After moving darker ones to the right\n List of disks\n";
//print out the sorted disk
print(disk, n);
//print the number of swap
auto stop = high_resolution_clock::now();

cout << "Number of Swaps is " << m << endl;

auto duration = duration_cast<microseconds>(stop - start);

    cout << "Time taken by function: "
         << duration.count() << " microseconds" << endl;
return 0;
}
```

OUTPUT

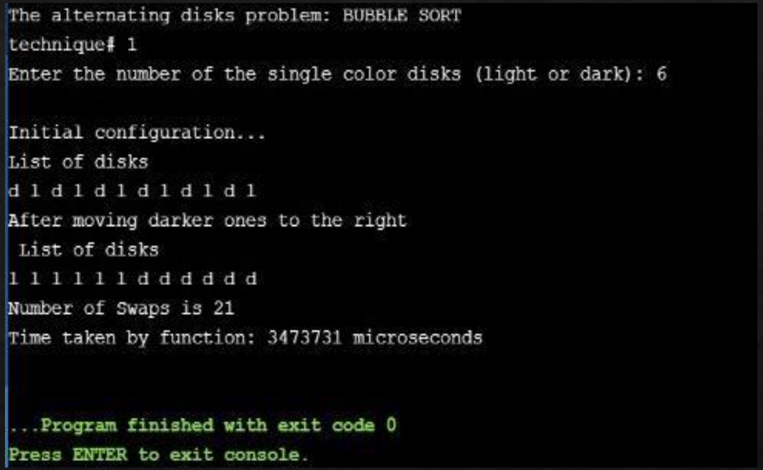


```
The alternating disks problem: BUBBLE SORT
technique# 1
Enter the number of the single color disks (light or dark): 4

Initial configuration...
List of disks
d l d l d l d l
After moving darker ones to the right
 List of disks
l l l l d d d d
Number of Swaps is 10



...Program finished with exit code 0
Press ENTER to exit console.
```

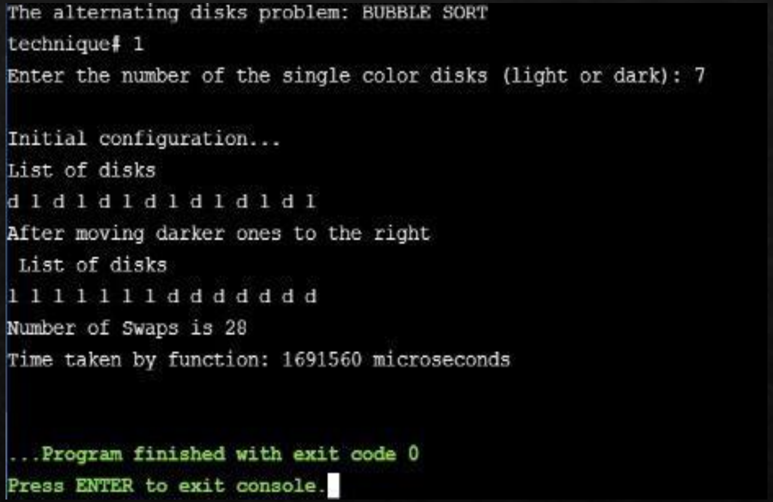
ANALYSING TIME EFFICIENCY



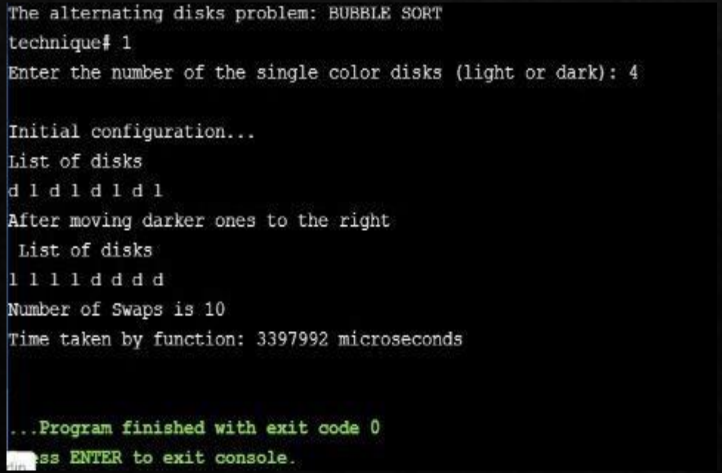
```
The alternating disks problem: BUBBLE SORT
technique# 1
Enter the number of the single color disks (light or dark): 3

Initial configuration...
List of disks
d l d l d l
After moving darker ones to the right
 List of disks
l l l d d d
Number of Swaps is 6
Time taken by function: 3206881 microseconds


...Program finished with exit code 0
Press ENTER to exit console.
```



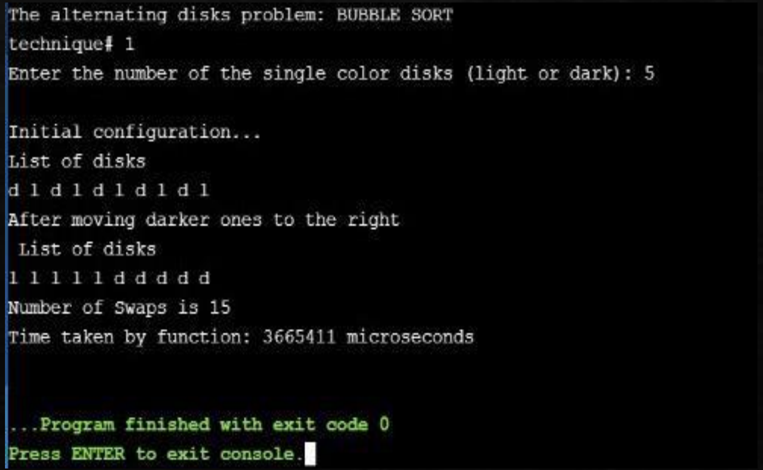
```
The alternating disks problem: BUBBLE SORT
technique# 1
Enter the number of the single color disks (light or dark): 6

Initial configuration...
List of disks
d l d l d l d l d l d l
After moving darker ones to the right
 List of disks
l l l l l l d d d d d d
Number of Swaps is 21
Time taken by function: 3473731 microseconds


...Program finished with exit code 0
Press ENTER to exit console.
```



```
The alternating disks problem: BUBBLE SORT
technique# 1
Enter the number of the single color disks (light or dark): 7

Initial configuration...
List of disks
d l d l d l d l d l d l d l
After moving darker ones to the right
 List of disks
l l l l l l l d d d d d d d
Number of Swaps is 28
Time taken by function: 1691560 microseconds


...Program finished with exit code 0
Press ENTER to exit console.
```



```
The alternating disks problem: BUBBLE SORT
technique# 1
Enter the number of the single color disks (light or dark): 4

Initial configuration...
List of disks
d l d l d l d l
After moving darker ones to the right
 List of disks
l l l l d d d d
Number of Swaps is 10
Time taken by function: 3397992 microseconds


...Program finished with exit code 0
 ss ENTER to exit console.
```



```
The alternating disks problem: BUBBLE SORT
technique# 1
Enter the number of the single color disks (light or dark): 5

Initial configuration...
List of disks
d l d l d l d l d l
After moving darker ones to the right
 List of disks
l l l l l d d d d d
Number of Swaps is 15
Time taken by function: 3665411 microseconds


...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion

Here is a simple and efficient (in fact, optimal) conclusion for this problem:
Starting with the first and ending with the last light disk, swap it with each of the i ($1 < i < n$) dark disks to the left of it. The ith iteration of the algorithm can be illustrated by the following diagram, in which is and Os correspond to the dark and light disks, respectively.

00..011..11010..10   00..0011..1110..10

i-1 i-1

The total number of swaps made is equal to 1 i = n(n + 1)/2.

THANKYOU

Done by

TEJASVI SHRIVASTAVA -1BM18IS117
SYED MAHMOOD N A -1BM18IS141