**SOURCE CODE**

**package** binary\_Search\_Tree;

**import** java.util.Scanner;

**import** java.util.concurrent.TimeUnit;

**public** **class** BSCS2\_Marasigan\_Binary

{

**public** **static** **void** main(String[] args)

{

Tools.*Head*();

Tools.*OptionLoop*();

Tools.*end*();

}

}

**class** Sort

{//I used quick sort because it's quick!

**static** **int**[] *array* = {5, 3, 4, 1, 2}; // just an initialization

Sort()

{

String decision = " ";

**if** (*Sorted*(*array*, 0, *array*.length-1))

{

decision = " Array is already sorted\n";

Tools.*Print*(decision, 0, decision.length(), 20);

} //String, start, end, speed

**else**

{

decision = " Sorting really quick. Please wait...\n";

Tools.*Print*(decision, 0, decision.length()-4, 20);

Tools.*Print*(decision, decision.length()-4, decision.length(), 1000);

*quickSort*(0, *array*.length-1);

}

}

**static** **void** quickSort(**int** start, **int** end)

{

**if** (start<end)

{

Tools.*PrintArray*(*array*, start, end);

//shows the elements that is currently being operated

}

**if** (start<end)

{

**int** spliter = *split*(start, end);

*quickSort*(start, spliter-1);

*quickSort*(spliter+1, end);

}

}

**static** **int** split(**int** start, **int** end)

{

**int** pivot = *array*[start];

**int** i = start+1, j=end;

**while** (i<j)

{//Ma'am's algorithm

**while**(pivot >= *array*[i] && i < end)

i++;

**while** (pivot < *array*[j])

j--;

**if** (i < j)

*Swap*(i, j);

}

**if** (*array*[j] < pivot)

*Swap*(j, start);

**return** j;

}

**static** **void** Swap(**int** index1, **int** index2)

{

**int** swap = *array*[index1];

*array*[index1] = *array*[index2];

*array*[index2] = swap;

}

**static** **boolean** Sorted(**int**[] array, **int** start, **int** end)

{

**for** (; start < end; start++)

{

**if** (array[start+1] < array[start] )

{

**return** **false**;//means it's not sorted yet

}

}

**return** **true**; //break

/\* Note: since this checker that is responsible for breaks is working properly

I decided to replace the algorithm a little to make use of its effectiveness\*/

}

}

**class** Search

{

**static** **int**[] *array* = { 5, 8, 6, 10, -1, 9}; //for testing purposes and initialization only

**static** String *answer* = " HISTORY: ";//records the events of recursion

Search(**int**[] array)

{

Search.*array* = array;

}

**static** **void** number(**int** searchNum)

{

String find = " Finding " + searchNum + "...\n";

Tools.*Print*(find, 0,find.length()-4, 10);

Tools.*Print*(find, find.length()-4,find.length(), 1000);

*binarySearch*(searchNum, 0, *array*.length);

}

**static** **void** binarySearch(**int** number, **int** start, **int** end)

{

**int** range = end - start; //limits the range of search

**int** add = range / 2;

**if** (range%2 == 1)

{//imitates the CEIL algorithm

add++;

}

**int** index = add+start-1;

/\*array starts at 0..n index must decrease by 1 because that's how array index works

this way elements, elements does not shift when array[index] < number to be searched \*/

**if** (index < 0)

{

*answer* += number +" not found";

//ends the method because there will be a time that the index is -1

//when both start and end becomes 0 during the recursion

//this happens when the number we are looking for is less than the first element

**return**;

}

Tools.*PrintArray*(*array*, index, index);

**if**(*array*[index] == number)

{//if element is found!

*answer* += number + " is found";

**return**;

}

*answer* += *array*[index] + ", ";

**if** ( range == 1 && *array*[index] != number)

{//if only 1 element remains and that element is not the number, (end)

*answer* += number +" not found";

**return**;

}

**if** (*array*[index] > number)

{//look from start to end of the range (left side)

*binarySearch*( number, start, index);

}

**else**

{//look from start to end of the range (right side)

*binarySearch*( number, index+1, end);

}

}

}

**class** Tools

{

**static** Scanner *in* = **new** Scanner(System.***in***);

//Main Tools

**static** **int**[] CreateArray()

{//this creates an array automatically without typing its size

String entry = *in*.nextLine();

String stringArray[];

stringArray = entry.split(" ");

/\*it uses a string and splits elements by spaces

which is then converted to an integer array\*/

**int** array[] = **new** **int**[stringArray.length];

**for** (**int** count = 0; count < array.length; count++)

{

array[count] = Integer.*parseInt*(stringArray[count]);

}

**return** array;

}

**static** **void** OptionLoop()

{

String phrase;

**char** choice = 'y';

**while** (*recurse*(choice))

{

phrase = "\n Please type the elements of the array: ";

Tools.*Print*(phrase, 0, phrase.length(), 10);

Sort.*array* = Tools.*CreateArray*();

**new** Sort();//checks if the array is sorted or not

**int**[] sorted\_array = Sort.*array*;

Tools.*PrintArray*(sorted\_array, 0, sorted\_array.length-1);//Prints w/ style

**new** Search(sorted\_array);

**while** (*recurse*(choice))

{

phrase = "\n Please type the element to be Searched: ";

Tools.*Print*(phrase, 0, phrase.length(), 20);

Search.*number*(*in*.nextInt());

Tools.*Print*(Search.*answer* + "\n", 0, (Search.*answer* + "\n").length(), 20);

phrase = " Do you want to search another element? [Y/N] ";

Tools.*Print*(phrase, 0, phrase.length(), 20);

choice = *in*.next().charAt(0);

Search.*answer* = " HISTORY: "; //resets the answer

}

phrase = " Do you want enter another array? [Y/N] ";

Tools.*Print*(phrase, 0, phrase.length(), 20);

choice = *in*.next().charAt(0);

*in*.nextLine();

}

}

**static** **boolean** recurse(**char** choice)

{

**boolean** recurse = **true**;

**if** (choice == 'y' || choice == 'Y')

**return** **true**;

**else** **if** (choice == 'n' || choice == 'N')

**return** **false**;

**return** recurse;

}

**static** **void** PrintArray(**int**[] array,**int** start, **int** end)

{

**int** count = 0;

System.***out***.print("\t");

**for** (; count < start; count++)

{

System.***out***.print("\t");

}

//prints roof depending on the elements within the range

System.***out***.print(".");

**for** (; count < end+1; count++)

{

**if**(count<end)

{

System.***out***.print("--------");

}

**else**

{

System.***out***.print("-------.\n");

}

}

//prints the array and the separation of the elements within the range

System.***out***.print("\t");

**for** (count = 0; count < start; count++)

{

System.***out***.print(array[count] + "\t");

}

System.***out***.print("|");

**for** (; count < end+1; count++)

{

System.***out***.print(array[count] + "\t");

}

System.***out***.print("|");

**for** (; count < array.length; count++)

{

System.***out***.print(array[count] + "\t");

}

System.***out***.println();

System.***out***.print("\t");

**for** (count= 0; count < start; count++)

{

System.***out***.print("\t");

}

System.***out***.print("'");

//prints floor

**for** (; count < end+1; count++)

{

**if**(count<end)

{

System.***out***.print("--------");

}

**else**

{

System.***out***.print("-------'");

}

}

Tools.*Print*("\n", 0, 1, 500); //just a small delay

}

**static** **void** Print(String text, **int** start, **int** end, **int** speed)

{

**try**

{

**for** (**int** count = start; count<end; count++)

{

System.***out***.print(text.charAt(count));

TimeUnit.***MILLISECONDS***.sleep(speed);

}

}

**catch** (Exception e) { }

}

//Design tools

**static** **void** Head()

{

String note = " .-[ Note ]----------------------------------------------------.\n"

+ " | Please enter all elements in one line only. Press space |\n"

+ " | after entering an element for entering the next element. |\n"

+ " | Just like: 5 4 3 9 2 1 90 -10 34 40 |\n"

+ " | Please follow the way of entering elements for correct |\n"

+ " | data processing -\_- | \n"

+ " '-------------------------------------------------------------'\n";

String arrow =" .\r\n"

+ " .\r\n"

+ " . '\r\n"

+ " .:'\r\n"

+ " ::.\r\n"

+ " :.::\r\n"

+ " ::::.\r\n"

+ " :::::\r\n"

+ " :::::\r\n"

+ " :::::\r\n"

+ " :::::\r\n"

+ " :::::\r\n"

+ " ..:::::..\r\n"

+ " ':::::'\r\n"

+ " ':'\r\n";

String bst = " .------------------------------------\\ /--------------------------------------.\n"

+ " | /\\|\\/\\ |\r\n"

+ " | .\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.. \_) (\_\_ |\r\n"

+ " | / W E L C O M E T O // \\\_ \_/ \_ |\r\n"

+ " | '==============================='' ) \\\_/ \\ |\r\n"

+ " | \\/\\ \\/ / |\r\n"

+ " | \_\_\_\_\_\_\_\_\_\_.\_\_ / \\ |\r\n"

+ " | \\\_\_\_\_\_\_ \\\_\_| \_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_.\_\_. /\\|\\/\\ /\\|\\/\\ |\r\n"

+ " | | - \_/ |/ \\\\\_\_ \\\\\_ \_\_ < | | \_) ( \_) (\_\_ |\r\n"

+ " | | \_\_ \\ | | \\/ \_\_ \\| | \\/\\\_\_\_ | \\\_ \_/ |\r\n"

+ " | |\_\_\_\_\_\_ /\_\_|\_\_\_| (\_\_\_\_ /\_\_| / \_\_\_\_| ) \\ \\ |\r\n"

+ " | \\/ \\/ \\/ \\/ \\/\\|\\/ /\\|\\/ |\r\n"

+ " | \_\_\_\_\_\_\_\_\_ .\_\_ /\\/\\ .\_.\_/ /\\ |\r\n"

+ " | / \_\_\_\_\_/ \_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_ | |\_\_ )/)/ | | )/ |\r\n"

+ " | \\\_\_\_\_\_ \\\_/ \_\_ \\\\\_\_ \\\\\_ \_\_ \\\_/ \_\_\_\\| | \\ |\_| |\r\n"

+ " | / \\ \_\_\_/ / \_\_ \\| | \\/\\ \\\_\_\_| Y \\ |-| |\r\n"

+ " | /\_\_\_\_\_\_\_ /\\\_\_\_ >\_\_\_\_ /\_\_| \\\_\_\_ >\_\_\_| / | | |\r\n"

+ " | \\/ \\/ \\/ \\/ \\/ |\_| |\n"

+ " '------------------------------------------- BY: VEM AIENSI MARASIGAN ^.\_.^ --'\r\n";

//prints note ( w/ manual delay manipulation)

*Print*(note, 0, 252, 20);

*Print*(note, 252, 276, 300);

*Print*(note, 276, note.length()-5, 20);

*Print*(note, note.length()-5, note.length(), 1000);

//Prints arrow

*Print*(arrow, 0, arrow.length(), 1);

//Printing Binary Search logo (Note\* this is manual tweaking)

**int** count = 0;

**for** (**int** line = 1; line<21; line++, count += 82)

{

*Print*(bst, count, 82\*line, 0);

*Print*(" ", 0, 1, 150);//delay

}

}

**static** **void** end()

{

String end = "\n\n\n\n\n\n\n\n\n\n\n\n"

+ "\t \* ,1111111. \*\r\n"

+ "\t 11111111111 .\r\n"

+ "\t 1111111111111\r\n"

+ "\t \* 1111111111111\r\n"

+ "\t 1111111111111\r\n"

+ "\t '11111111111'\r\n"

+ "\t '1111111' \*\r\n"

+ "\t |\\\_\_\_/| -\r\n"

+ "\t ) ( . '\r\n"

+ "\t =\\ /=\r\n"

+ "\t )===( \*\r\n"

+ "\t / \\\r\n"

+ "\t | |\r\n"

+ "\t / \\\r\n"

+ "\t \\ /\r\n"

+ "\t \_/\\\_/\\\_/\\\_\_ \_/\_/\\\_/\\\_/\\\_/\\\_/\\\_/\\\_/\\\_/\\\_/\\\_\r\n"

+ "\t | | | |( ( | | | | | | | | | |\r\n"

+ "\t | | | | ) ) | | | THANK YOU PO | | |\r\n"

+ "\t | | | |(\_( | | | | | | | | | |\r\n"

+ "\t | | | | | | BY: VEM AIENSI MARASIGAN |\r\n"

+ "\t | | | | | | | | | | | | | | |\n"

+ "\n\n\n\n\n\n\n\n\n";

String credits = "\t ASCII arts are from: \n"

+ "\t http://user.xmission.com/~emailbox/ascii\_cats.html\n"

+ "\t and patorjk.com/software/taag";

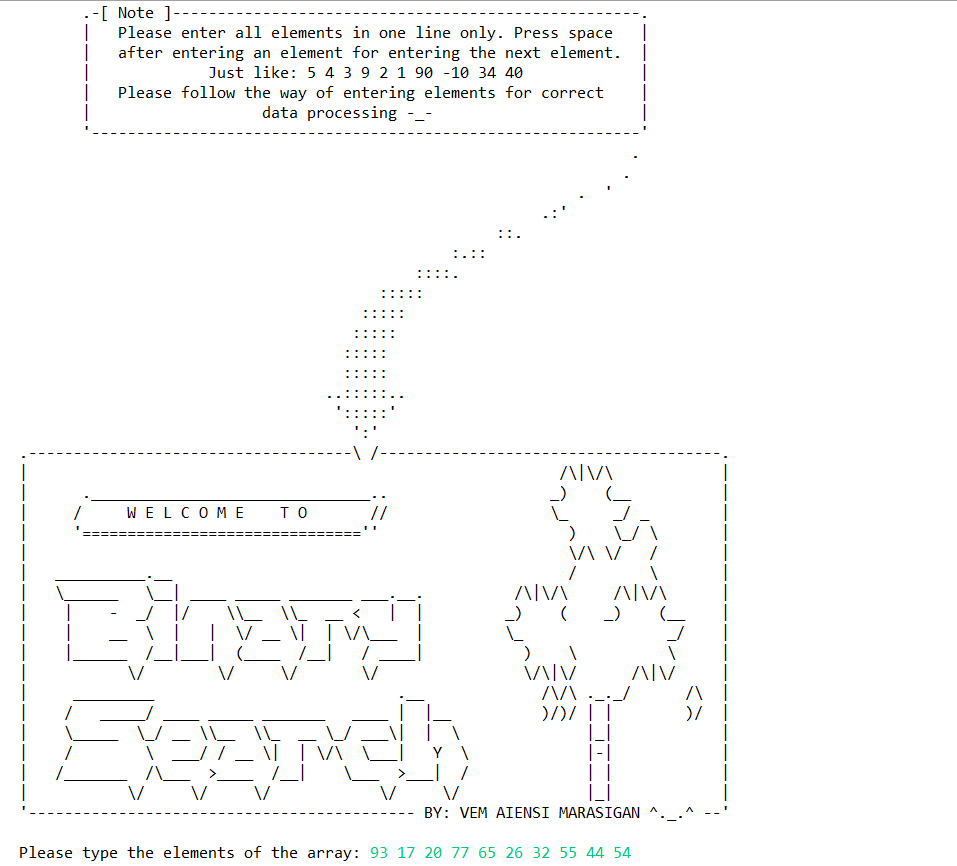
*Print*(end, 0, end.length(), 5);

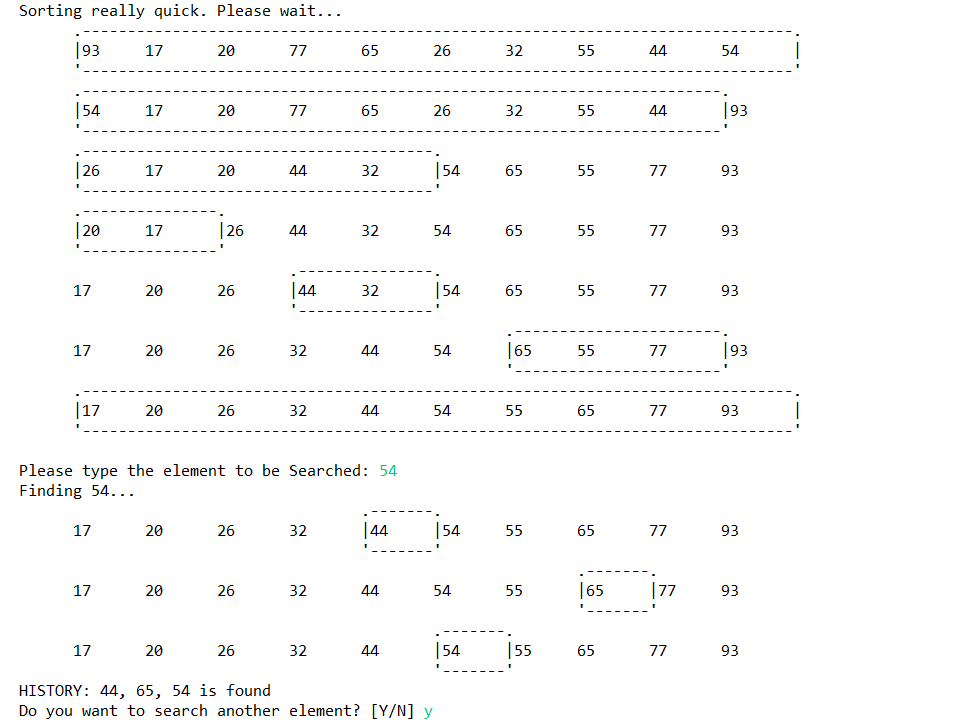
*Print*(credits, 0, credits.length(), 30);

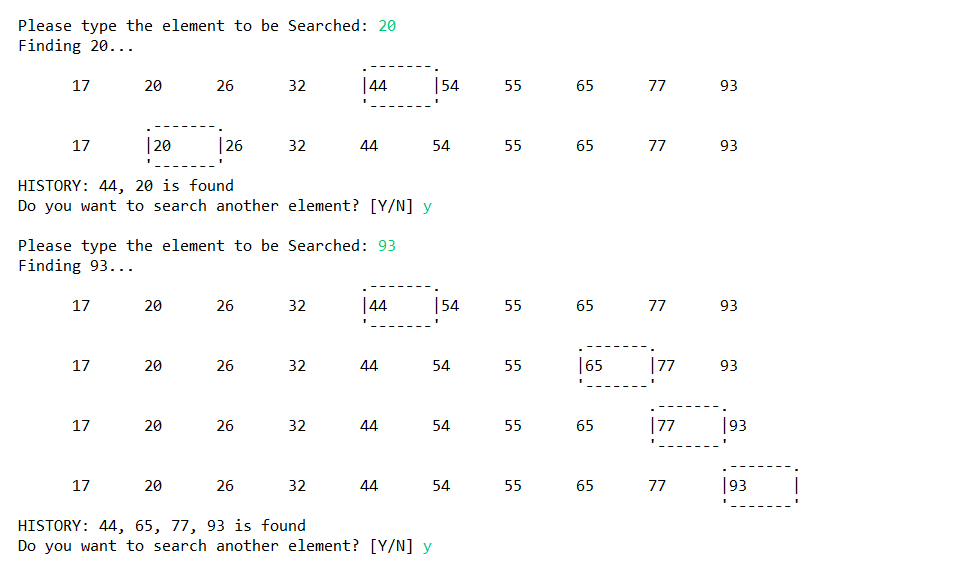
}

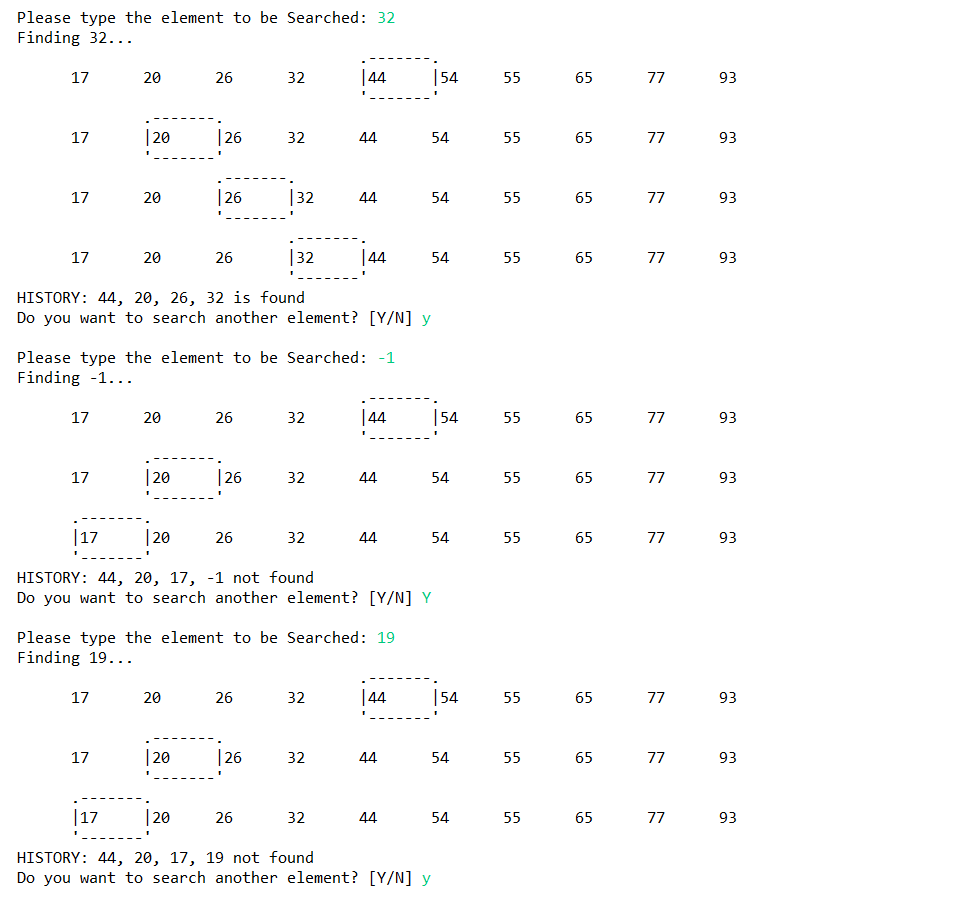
}

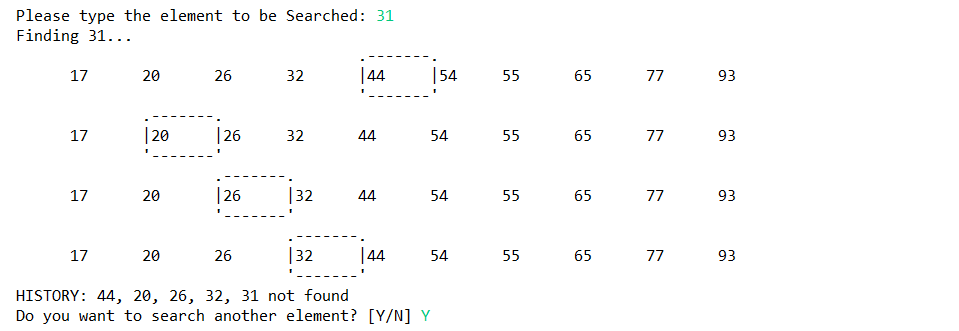
**SCREENSHOTS OF OUTPUT**

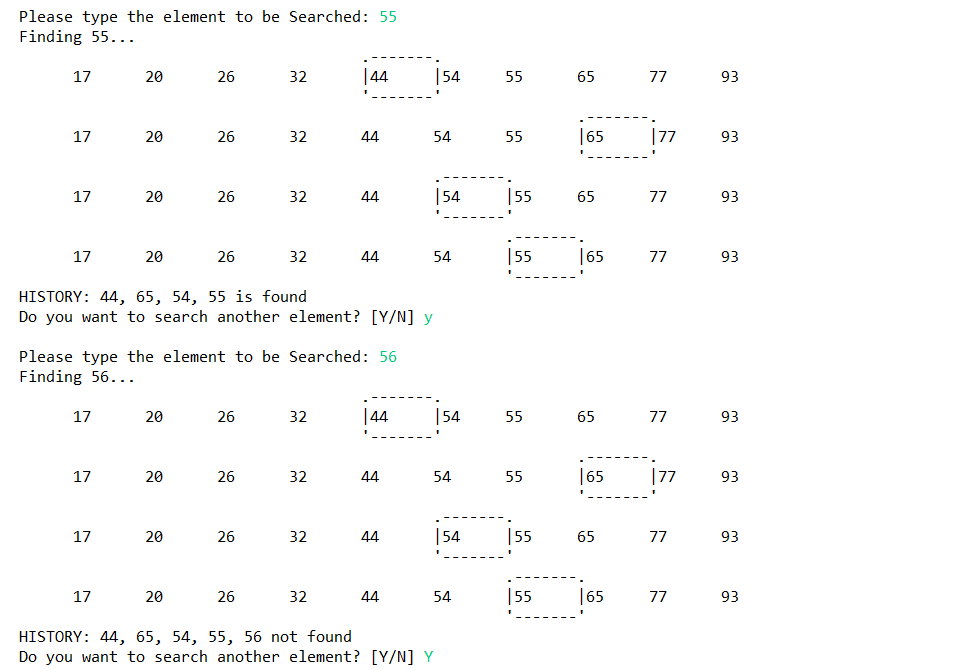
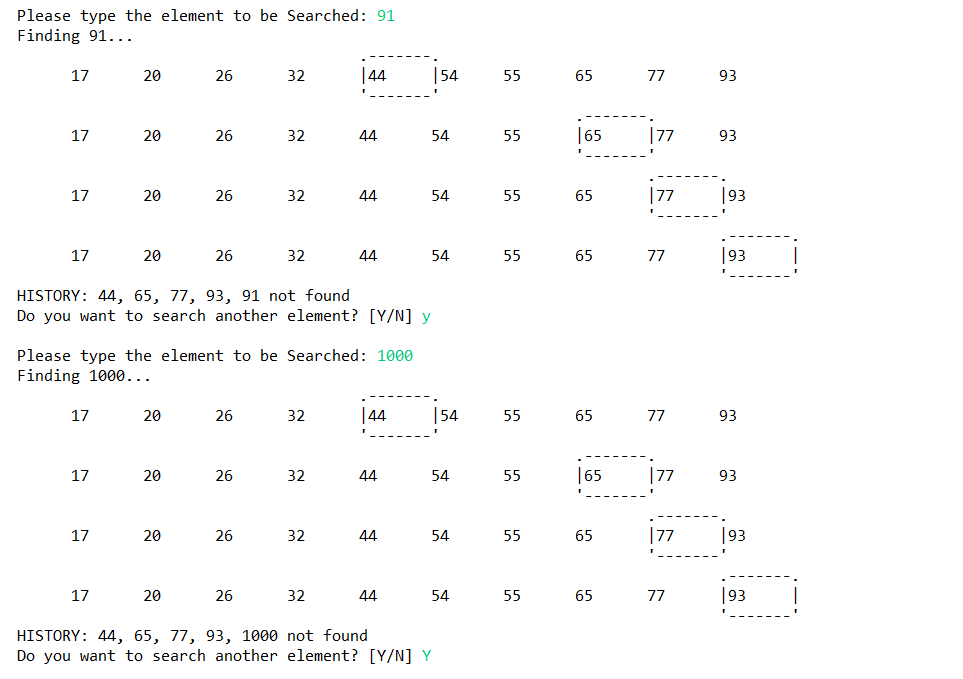
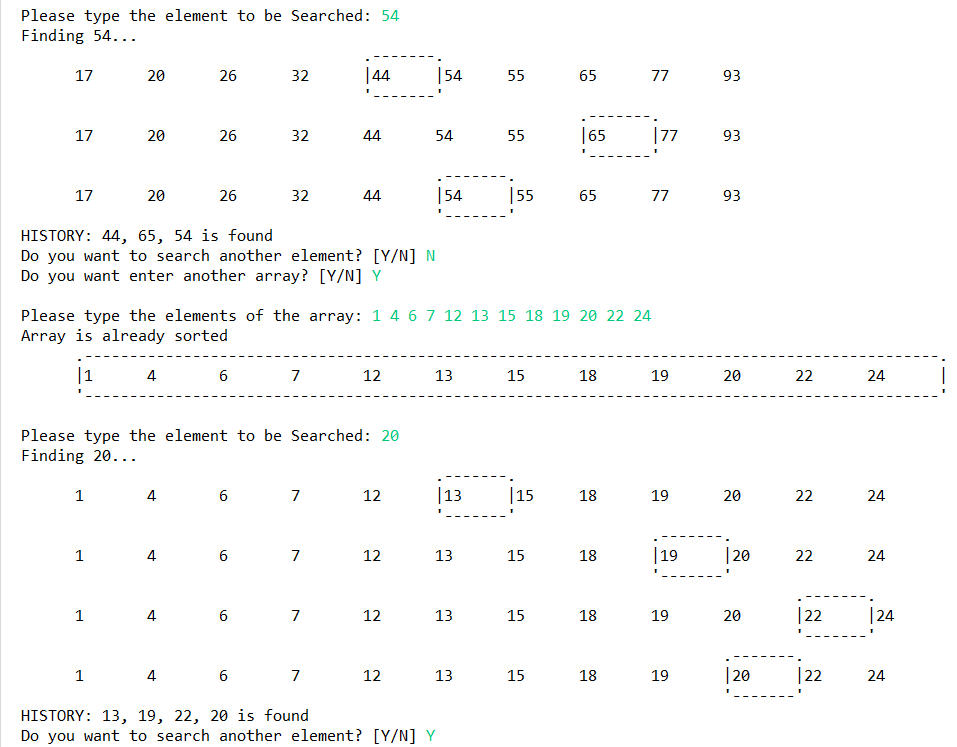
****

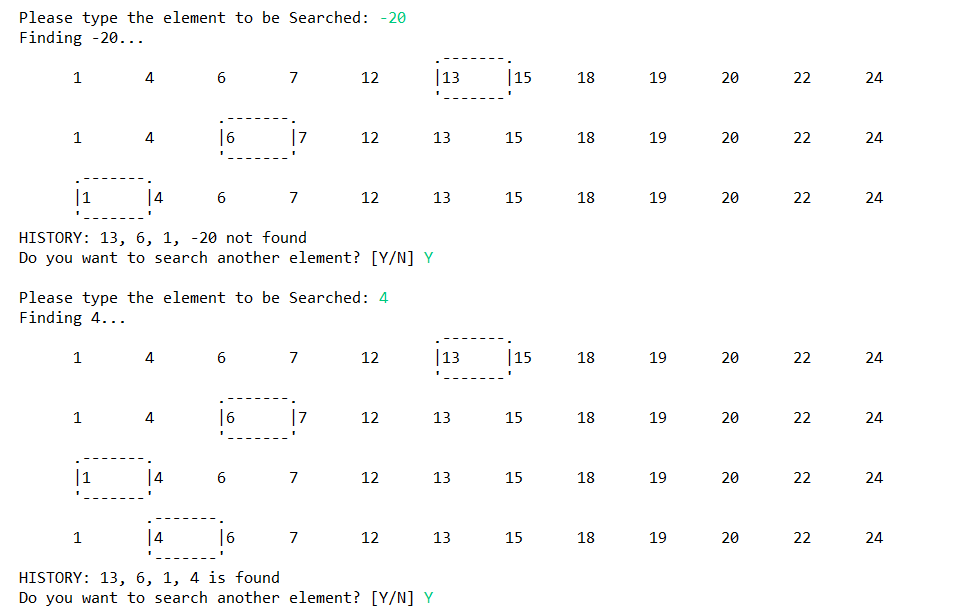
****

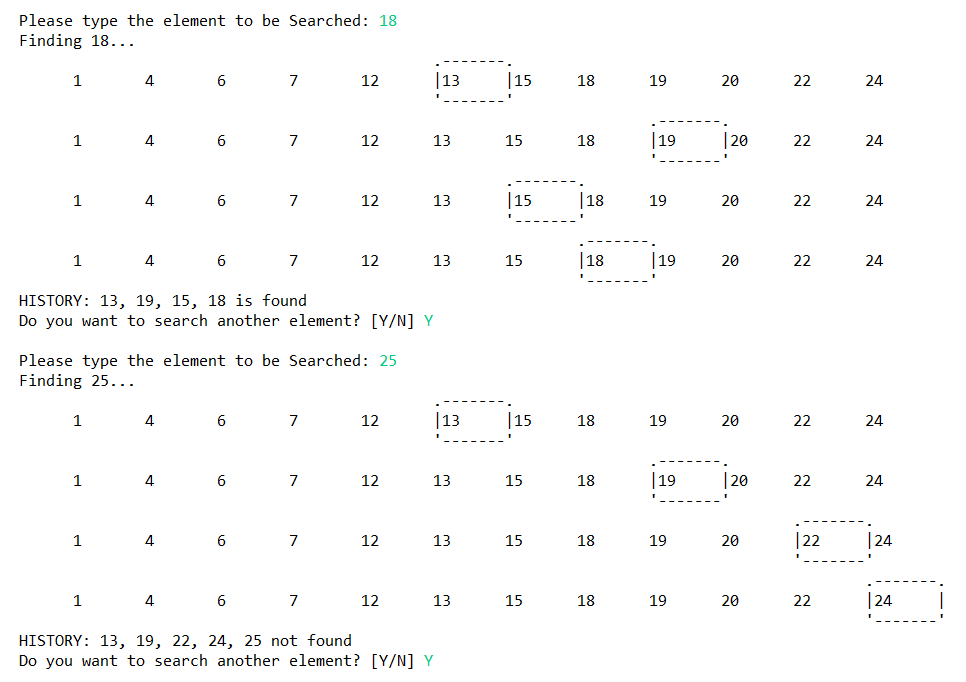
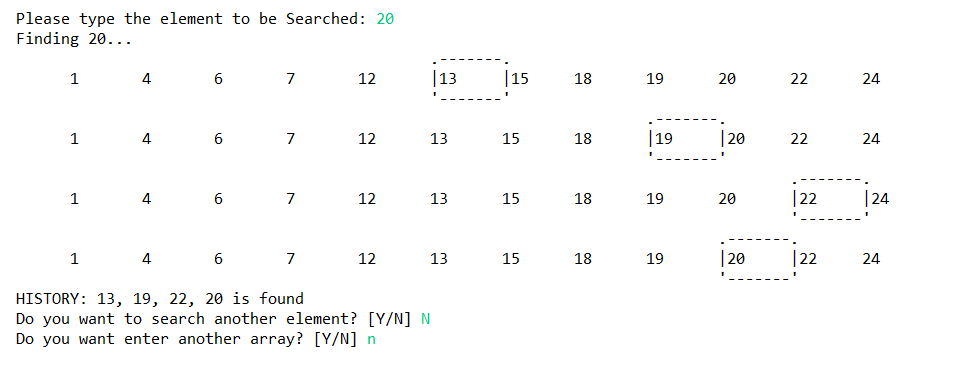
****

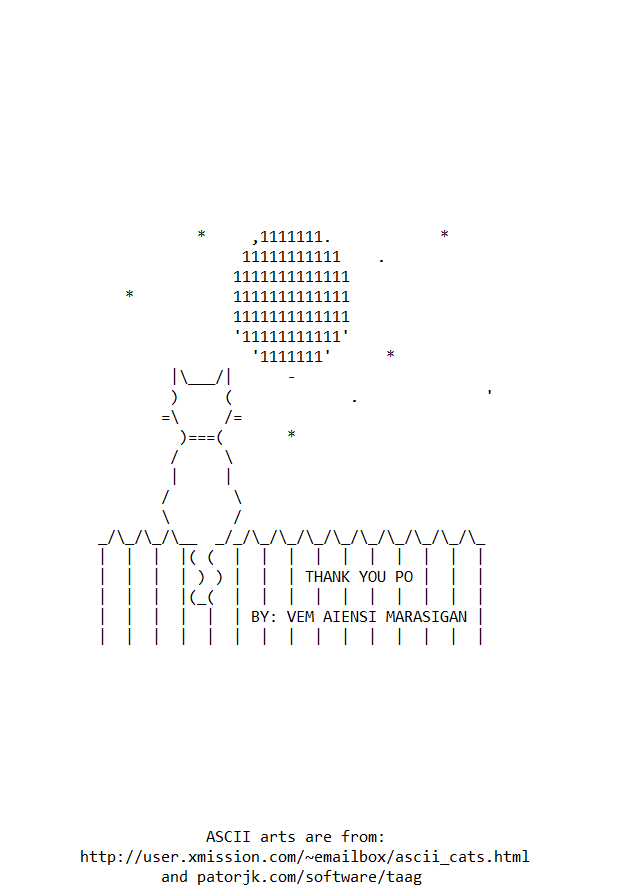
****







Running Video: <https://drive.google.com/file/d/1DzxErJ8a066_JL9U9sr0F31mtvK75bSG/view?usp=sharing>