```java
/**
 * @file Test.java
 * @author Jackson York
 * @brief Test functions with sample main() for Exam 1 for Data Structures and Algori
thms
 * @date 2021-02-25
 */

public class Test {

    public static void testq1() {
        Question1.Array<Integer> a = new Question1.Array<Integer>(new Integer[] { 5,
2, 3, 1, 4 }, 5);
        a.println();
        a.remove_at(2);
        a.println();
        a.insert_at(2, 1);
        a.println();
        a.filter((Integer x) -> {
            return x % 2 == 0;
        }).println();
        System.out.println("a.size()\t\t\t" + a.size());
        System.out.println("a.index_of(4)\t\t\t" + a.index_of(4));
        System.out.println("a.index_of(7)\t\t\t" + a.index_of(7));
        System.out.println("a.is_empty()\t\t\t" + a.is_empty());
        System.out.print("a.filter((x)->{x%2==0})\t\t");
        a.filter((Integer x) -> {
            return x % 2 == 0;
        }).println();
        Question1.Array<Integer> temp = new Question1.Array<Integer>(3);
        temp.print();
        System.out.println(".is_empty()\t" + temp.is_empty());
        System.out.println("a.is_equal(temp)\t\t" + a.is_equal(temp));
        temp = new Question1.Array<Integer>(new Integer[] { 5, 2, 1, null, 4 }, 5);
        System.out.print("a.is_equal(");
        temp.print();
        System.out.println(")\t" + a.is_equal(temp));
        temp = a.shuffle();
        temp.println();
        temp.sort().println();
        temp.slice(1, 3).println();
    }

    private static void print(int[] A) {
        System.out.print("[" + A[0]);
        for (int i = 1; i < A.length; i++) {
            System.out.print(", " + A[i]);
        }
        System.out.print("]");
    }
```

```
    public static void testq2() {
        int n = 4;
        int A[] = new int[] { 1, 3, 5, 7 };
        System.out.print("A = ");
        print(A);
        System.out.println();
        for (int i = -3; i < 10; i++) {
            System.out.print("is_k_bit_int_not_in_A(" + i + ", A, " + n + ")  \t-
> ");
            System.out.println(Question2.is_k_bit_int_not_in_A(i, A));
        }
    }

    public static void testq3() {
        int test[] = { 2, 3, 1, -4, -5 };
        System.out.print("find_unformable(");
        print(test);
        System.out.print(") -> \t");
        System.out.println(Question3.find_unformable(test));
    }

    public static void testq4() {
        int pivot = 2;
        int A[] = { 5, 3, 2, 4, 1 };
        System.out.print("rearrange_around_pivot(A, pivot)" + "\n\tA = ");
        print(A);
        System.out.print(",\tpivot = " + pivot + "\t-> ");
        Question4.rearrange_around_pivot(A, pivot);
        pivot = 3;
        print(A);
        System.out.print("\n\tA = ");
        print(A);
        System.out.print(",\tpivot = " + pivot + "\t-> ");
        Question4.rearrange_around_pivot(A, pivot);
        pivot = 4;
        print(A);
        System.out.print("\n\tA = ");
        print(A);
        System.out.print(",\tpivot = " + pivot + "\t-> ");
        Question4.rearrange_around_pivot(A, pivot);
        print(A);
        System.out.println();
    }

    public static void testq5() {
        Question5.String this_str = new Question5.String(new Character[] { 'F', 'o',
'o' });
        Question5.String other_str = new Question5.String(new Character[] { ' ', ' ',
 ' ', 'B', 'a', 'r' });
```

```
        System.out.print("this        -> \t");
        this_str.print();
        System.out.println();
        System.out.print("other       -> \t");
        other_str.print();
        System.out.println();
        System.out.println("strlen()    -> \t" + this_str.strlen());
        System.out.print("strlwr()    -> \t");
        this_str.strlwr().print();
        System.out.println();
        System.out.print("strlwr()    -> \t");
        this_str.strupr().print();
        System.out.println();
        System.out.print("other       -> \t");
        other_str.print();
        System.out.println();
        System.out.print("strcat(o)   -> \t");
        this_str.strcat(other_str).print();
        System.out.println();
        System.out.print("other       -> \t");
        other_str.print();
        System.out.println();
        System.out.println("strcpy(o)   -> \t" + this_str.strcpy(other_str));
        System.out.print("other       -> \t");
        other_str.print();
        System.out.println();
        System.out.println("o.strcpy(t) -> \t" + other_str.strcpy(this_str));
        System.out.print("this        -> \t");
        this_str.print();
        System.out.println();
        System.out.println("strcmp(o)   -> \t" + this_str.strcmp(other_str));
        System.out.println("strcmp(t)   -> \t" + this_str.strcmp(this_str));
        System.out.println("strcmp(\" \") -
> \t" + this_str.strcmp(new Question5.String(new Character[] { ' ' })));
    }

    public static void main(String[] args) {
        System.out.println("\n\nQUESTION 1 TEST OUTPUT");
        testq1();
        System.out.println("\n\nQUESTION 2 TEST OUTPUT");
        testq2();
        System.out.println("\n\nQUESTION 3 TEST OUTPUT");
        testq3();
        System.out.println("\n\nQUESTION 4 TEST OUTPUT");
        testq4();
        System.out.println("\n\nQUESTION 5 TEST OUTPUT");
        testq5();
    }
}
```

```java
/**
 * @file Question1.java
 * @author Jackson York
 * @brief Question 1 on Exam 1 for Data Structures and Algorithms
 * @date 2021-02-25
 */

import java.lang.Math;
public class Question1 {
    public interface FunctionalInterface<T> {
        boolean call(T value);
    }

    public static class Array<T extends Comparable<T>> {

        private int length;
        private T[] data;

        // runtime = 2
        @SuppressWarnings("unchecked")
        public Array(int length) {
            this.data = (T[]) new Comparable[length]; // 1
            this.length = length; // 1
        };

        // runtime = 3n+4
        @SuppressWarnings("unchecked")
        public Array(T[] arr, int n) {
            this.data = (T[]) new Comparable[n]; // 1
            for (int i = 0; i < n; i++) // 2n+2
            {
                this.data[i] = arr[i]; // n
            }
            this.length = n; // 1
        };

        // runtime = 3n+4
        @SuppressWarnings("unchecked")
        public Array(T arr[]) {
            this.data = (T[]) new Comparable[arr.length];
            for (int i = 0; i < arr.length; i++) {
                this.data[i] = arr[i];
            }
            this.length = arr.length;
        };

        /**
         * @note runtime = 3n+3 = O(n)
         *
         * @brief is the array all null values
```

```
 *
 * @return whether the array is empty
 */
public boolean is_empty() {
    for (int i = 0; i < length; i++) // 2n+2
        if (data[i] != null) // n
            return false;
    return true; // 1
};

/**
 * @note runtime = 3n+4 = O(n)
 *
 * @brief are all elements in each array equal?
 *
 * @param other Array to check
 * @return whether all elements in this equal all elements in other
 */
public boolean is_equal(Array<T> other) {
    if (this.length != other.length)
        return false;
    for (int i = 0; i < this.length; i++)
        if (this.data[i] != other.data[i])
            return false;
    return true;
};

/**
 * @note runtime = 1 = O(1)
 *
 * @brief replaces the value at index with value.
 *
 * @param index at which to insert
 * @param value to insert
 * @return (void)
 */
public void insert_at(int index, T value) {
    this.data[index] = value;
};

/// runtime = 1, O(1)
public T at(int index) {
    return this.data[index];
};

/**
 * @note runtime = 7n+n*runtime(f)+10 = O(n) if f~O(1) = O(n*O(f)) else
 *
 * @brief filters elements by the result of f
 *
```

```java
     * @param f test function
     * @return Array<T>* of elements where f(T value) returned true.
     */
    @SuppressWarnings("unchecked")
    public Array<T> filter(FunctionalInterface<T> f) {
        T[] arr = (T[]) new Comparable[length]; // 1
        int j = 0; // 1
        for (int i = 0; i < length; i++) // 2n+2
        {
            if (f.call(this.data[i])) // n*runtime(f)
            {
                arr[j] = this.data[i]; // n
                j++; // n
            }
        }
        Array<T> out = new Array<T>(arr, j); // 1 + (3n+4)
        return out; // 1
    }

    /**
     * @note runtime = 1 = O(1)
     *
     * @brief returns the size of the array.
     *
     * @return int size
     */
    public int size() {
        return length;
    }

    /**
     * @note runtime = 1 = O(1)
     *
     * @brief sets the value at index to null.
     *
     * @param index value to remove.
     */
    public void remove_at(int index) {
        this.data[index] = null; // 1
    }

    /**
     * @note runtime = 1.5n^2 + 4.5n + 2 = O(n^2)
     *
     * @brief removes all values that are repeats after their first appearance.
     *
     */
    public void remove_duplicates() {
        for (int i = 0; i < length; i++) // 2n+2
        {
```

```
                if (this.data[i] != null) // n | worst case scenario is a list with n
o duplicates
                {
                    for (int j = i + 1; j < length; j++) // n*(2[(n-
1)/2]+3) | since j=i+1, the average length of the
                                                         // loop is (n-1)/2
                    {
                        if (this.data[i] == this.data[j]) // n*(n-1)/2
                        {
                            // can happen a max of n-
1 times because data[i]!= null, so can only act on
                            // an
                            // element after data[0] once.
                            // This max does not matter, however, because reaching it
 would decrease the
                            // total runtime by an order of magnitude(data[i]!= null)
.
                            // therefore, the worst case is a list with no repeats.
                            this.data[j] = null; // 0
                        }
                    }
                }
            }
        }

        /**
         * @note runtime = 4n + 4 = O(n)
         *
         * @brief prints the array inline
         *
         */
        public void print() {
            if (this.length == 0) // 1
            {
                System.out.print("[]");
                return;
            }
            System.out.print("[" + this.data[0]); // 2
            for (int i = 1; i < length; i++) // 2(n-1) + 2
            {
                System.out.print(", " + this.data[i]); // 2n
            }
            System.out.print("]"); // 1
        };

        /**
         * @note runtime = 4n + 5 = O(n)
         *
         * @brief prints the array with a newline at the end
         *
```

```
         */
        public void println() {
            print();
            System.out.println();
        }

        /// runtime = 4nlogn + 13n - 4logn + 6, O(nlogn)
        private void quicksort(T[] array, int low_index, int high_index) {
            if (low_index < high_index) // 2n-1 | n-1 for each split, n for each end
            {
                int pivot_index = partition(array, low_index, high_index); // (n-
1)*(4(hi-lo) + 9) = (n - 1) * (4 log n + 9)
                quicksort(array, low_index, pivot_index); // n-1
                quicksort(array, pivot_index + 1, high_index); // n-1
            }
        }

        /// runtime = 4n + 8 | n = hi - lo
        private int partition(T[] array, int low_index, int high_index) {
            T pivot_value = array[(high_index + low_index) / 2]; // 3
            int i = low_index - 1; // 2
            int j = high_index + 1; // 2
            while (true) {
                do
                    i++; // n/2
                while (array[i].compareTo(pivot_value) < 0); // n/2
                do
                    j--; // n/2
                while (array[j].compareTo(pivot_value) > 0); // n/2

                if (i >= j) // n/2
                    return j; // 1

                T temp = array[i]; // n/2
                array[i] = array[j]; // n/2
                array[j] = temp; // n/2
            }
        }

        /**
         * @note runtime = 4nlogn + 16n - 4logn + 12 = O(nlogn)
         *
         * @brief sorts the array using the quicksort algorithm
         *
         * @return sorted array
         */
        public Array<T> sort() {
            Array<T> out = new Array<T>(this.data, this.length); // 3n+5
            quicksort(out.data, 0, out.length - 1); // 4nlogn + 13n - 4logn + 6, O(nl
ogn)
```

```
    return out; // 1
};

/**
 * @note runtime = 3n+3 = O(n)
 *
 * @brief searches for a value and returns the index.
 *
 * @param value to find
 * @return int
 */
int index_of(T value) {
    for (int i = 0; i < this.length; i++) // 2n+2
        if (this.data[i].compareTo(value) == 0) // n
            return i;
    return -1; // 1
};

/**
 * @note runtime = 14n + 12 = O(n)
 *
 * @brief shuffles the values in this array.
 *
 * @return Array<T>* containing the randomly shuffled values
 */
Array<T> shuffle() {
    Array<T> out = new Array<T>(this.data, this.length); // 3n+5
    int seed = (int) (System.currentTimeMillis()%Integer.MAX_VALUE); // 4
    for (int i = 0; i < out.length - 2; i++) // 2n+2
    {
        int random = (int)(Math.random() * seed)%(out.length-i) + i; // 6n
        T temp = out.data[i]; // n
        out.data[i] = out.data[random]; // n
        out.data[random] = temp; // n
    }
    return out; // 1
};

/**
 * @note runtime = 4n + 7
 * @brief Returns the slice of the array in the range [min, max)
 *
 * @param min lowest index to include (inclusive)
 * @param max highest index to include (exclusive)
 * @return Array<T>* containing the values of this in the range [min, max)
 */
Array<T> slice(int min, int max) {
    Array<T> out = new Array<T>(max - min); // 4
    for (int i = min; i < max; i++) // 2n+2
        out.data[i - min] = this.data[i]; // 2n
```

```
            return out; // 1
        };

    }

}
```

```java
/**
 * @file Question2.java
 * @author Jackson York
 * @brief Question 2 on Exam 1 for Data Structures and Algorithms
 * @date 2021-02-25
 */

public class Question2 {
    /**
     * @note runtime = 3 * n + 28 = O(n)
     *
     * @brief Given an array of n positive odd integers, each represented with
     *        k=|`logn`|+1 bits, write an O(n)-time method for finding whether an
     *        integer is a k-bit integer not in A.
     *
     * @param x The number that must be k-bits and not in A
     * @param A The array containing only positive odd k-bit integers
     */
    public static boolean is_k_bit_int_not_in_A(int x, int A[]) {
        // int k = ceil(log2((double)n)) + 1; // k=|`logn`|+1 bits
        int k = (int) Math.ceil(Math.log((double) A.length) / Math.log(2)) + 1; // k=
|`log_2 n`|+1 bits
        if (k < 1)
            k = 1;
        // std::cout << "k = " << k;
        int num_of_bits_in_x = (int) (Math.log((double) x) / Math.log(2)) + 1;
        if (num_of_bits_in_x < 1)
            num_of_bits_in_x = 1;
        // std::cout << "\tb = " << num_of_bits_in_x << "\t";

        // I'm choosing to throw out negatives because the professor said in class th
at,
        // for this problem, (0=0, 1=1, 2=10, 3=11, ... )
        // This indicates that 2's complement notation (-2=10,-
1=11, 0 = 0, 1=01, ...)
        // is invalid (and thus negative values are invalid).
        if (num_of_bits_in_x > k || x < 0)
            return false; // is not "a positive k-bit int"
        if (x % 2 == 0)
            return true; // is not odd, therefore is "a positive k-
bit integer not in A"
        for (int i = 0; i < A.length; i++)
            if (x == A[i])
                return false; // is not "not in A"
        return true; // is "a positive k-bit integer not in A"
    }
}
```

```java
/**
 * @file Question3.java
 * @author Jackson York
 * @brief Question 3 on Exam 1 for Data Structures and Algorithms
 * @date 2021-02-25
 */

import java.lang.Math;

public class Question3 {
    /**
     * @note runtime = 6 * n + 7 = O(n)
     *
     * @brief This algorithm relies on the understanding that if the maximum value
     *        in an array is added to itself, that is the highest value achievable
     *        by summing values in that array.
     *
     * @param A Array with arbitrary integers
     * @return int that cannot be formed from the sum of any two values in A
     */
    public static int find_unformable(int[] A) {
        if (A.length < 1)
            return 1; // 1 cannot be formed from the sum of two non-existant numbers.
        int max = A[0];
        for (int i = 1; i < A.length; i++) // iterate over all elements after A[0]
            if (Math.abs(A[i]) > Math.abs(max))
                max = A[i]; // set max if new max is found
        return Math.abs(max) * 2 + 1; // return 1 more than highest possible sum
    }
}
```

```java
/**
 * @file Question4.java
 * @author Jackson York
 * @brief Question 4 on Exam 1 for Data Structures and Algorithms
 * @date 2021-02-25
 */

public class Question4 {
    /**
     * @note runtime = 8n+2 = O(n)
     *
     * @brief Recursively rearranges A so that elements larger than k are to the
     *        right of other elements.
     *
     * @param A Array to be rearranged
     * @param k value around which to pivot
     */
    public static void rearrange_around_pivot(int[] A, int k) {
        recursive_call(A, 0, A.length, k);
    }

    private static void recursive_call(int[] A, int i, int n, int k) {
        if (i == n) // every call of this function moves n and i closer together.
            return; // this stops the calls when all elements have been switched or s
kipped.

        if (A[i] > k) { // switch {n-1}th element and {i}th element
            int temp = A[i];
            int n_1 = n - 1;
            A[i] = A[n_1];
            A[n_1] = temp;
            recursive_call(A, i, n_1, k); // moves n closer to i
        } else { // skip {i}th element
            recursive_call(A, i + 1, n, k); // moves i closer to n
        }
    }
}
```

```java
/**
 * @file Question5.java
 * @author Jackson York
 * @brief Question 5 on Exam 1 for Data Structures and Algorithms
 * @date 2021-02-25
 */

public class Question5 {
    // unsafe flag
    private static class unsafe {
    };

    private static final unsafe UNSAFE = null;

    public static class String {
        Question1.Array<Character> values = null;

        /// runtime = 1
        private String(Question1.Array<Character> values, unsafe u) {
            // unsafe, uses given value array without copying
            this.values = values;
        };

        /// runtime = 3n+4
        public String(Character[] char_array) { // template allows entry of only Char
acter[].
            this.values = new Question1.Array<Character>(char_array.length);
            for (int i = 0; i < char_array.length; i++) {
                this.values.insert_at(i, char_array[i]);
            }
        };

        /// runtime = 8n+11
        public String(Question1.Array<Character> values) {
            // copies all values by filtering with a condition of true.
            this.values = values.filter((Character __) -> {
                return true;
            });
        };
        /// runtime = 2

        /**
         * @note runtime = 2 = O(1)
         *
         * @brief length of the string
         *
         * @return int the length of the string
         */
        int strlen() {
            return values.size();
```

```
        }

        /**
         * @note runtime = 10n + 10 = O(n)
         *
         * @brief converts all uppercase characters to their lowercase equivalents
         *
         * @return String* converted string
         */
        String strlwr() {
            // setting this_len causes an order-
n speedup as size() would be called in the
            // for loop.
            int this_len = this.values.size();
            Question1.Array<Character> lower_arr = new Question1.Array<Character>(thi
s_len);
            for (int i = 0; i < this_len; i++) {
                // this can save up to 3n in runtime at face value. I could use [[gnu
::pure]] to
                // cause
                // this behavior at compile time, at the expense of explicitness.
                Character c = this.values.at(i);
                // ant math is calculated at compile time. ('a' - 'A')=32 is purely f
or
                // readability.
                // triconditional helps with caching.
                lower_arr.insert_at(i, (char) (c + ((c >= 'A' && c <= 'Z') ? ('a' - '
A') : 0)));
            }
            // using unsafe version because upper_arr is otherwise inaccessible.
            return new String(lower_arr, UNSAFE);
        }

        /**
         * @note runtime = 10n + 10 = O(n)
         *
         * @brief converts all lower case characters to their uppercase equivalents.
         *
         * @return String* converted string
         */
        String strupr() {
            // see strlwr() for comments, these are identical outside of logic.
            int this_len = this.values.size();
            Question1.Array<Character> upper_arr = new Question1.Array<Character>(thi
s.values.size());
            for (int i = 0; i < this_len; i++) {
                Character c = this.values.at(i);
                upper_arr.insert_at(i, (char) (c - ((c >= 'a' && c <= 'z') ? ('a' - '
A') : 0)));
            }
```

```
            return new String(upper_arr, UNSAFE);
        }

        /**
         * @note runtime = 4m+5p + 13 | m := this_len, p := other_len = 9n + 13 | n :
=
         *       (4m + 5p) / 9 = (weighted) average length of input strings = O(n)
         *
         * @brief appends the given string to the end of this string
         *
         * @param other string to append
         * @return String* containing the concatenated string
         */
        String strcat(String other) {
            // setting this_len, other_len causes an order-
n speedup as size() would be
            // called in the for loop.
            int this_len = this.values.size();
            int other_len = other.values.size();
            Question1.Array<Character> concat_arr = new Question1.Array<Character>(th
is_len + other_len);
            for (int i = 0; i < this_len; i++)
                concat_arr.insert_at(i, this.values.at(i));
            for (int i = 0; i < other_len; i++)
                concat_arr.insert_at(this_len + i, other.values.at(i));
            return new String(concat_arr, UNSAFE);
        }

        /**
         * @note runtime = 4n + 7 = O(n)
         *
         * @brief copies this string into the destination string
         *
         * @param dest_str the destination string
         * @return boolean whether the copy succeeded
         */
        boolean strcpy(String dest_str) {
            int this_len = this.values.size(); // 2
            if (this_len > dest_str.values.size()) // 2
                return false; // cannot copy string to other.
            for (int i = 0; i < this_len; i++) // 2n+2
                dest_str.values.insert_at(i, this.values.at(i)); // 2n
            return true; // 1
        }

        /**
         * @note runtime = 10n + 13 = O(n)
         *
         * @brief compares this with other, returning 0 if equal, 1 if greater than,
-1
```

```
         *        if less than
         *
         * @param other string to compare
         * @return int ( == 0; > 1; < -1 )
         */
        int strcmp(String other) {
            int this_len = this.values.size();
            int other_len = other.values.size();
            for (int i = 0; i < this_len && i < other_len; i++)
                if (this.values.at(i) < other.values.at(i))
                    return -1;
                else if (this.values.at(i) > other.values.at(i))
                    return 1;
            return this_len == other_len ? 0 : (this_len > other_len ? 1 : -1);
        }

        public void print() {
            int this_len = this.values.size();
            System.out.print("\"");
            for (int i = 0; i < this_len; i++)
                System.out.print(this.values.at(i));
            System.out.print("\"");
        }
        // friend std::ostream &operator<<(std::ostream &o, String s);
    };
}
```

```cpp
/**
 * @file Test.cpp
 * @author Jackson York
 * @brief Test functions with sample main() for Exam 1 for Data Structures and Algori
thms
 * @date 2021-02-25
 */

#include "./Question1.cpp"
#include "./Question2.cpp"
#include "./Question3.cpp"
#include "./Question4.cpp"
#include "./Question5.cpp"

void testq1()
{
    Array<int> *a = new Array<int>(new int[5]{5, 2, 3, 1, 4}, 5);
    std::cout << "a\t\t\t\t";
    a->println();
    a->remove_at(2);
    std::cout << "a->remove_at(2)\t\t\t";
    a->println();
    a->insert_at(2, 1);
    std::cout << "a->insert_at(2,1)\t\t";
    a->println();
    std::cout << "a->filter()\t\t\t";
    a->filter([](int x) -> bool { return x % 2 == 0; })->println();
    std::cout << "a->size()\t\t\t" << a->size() << std::endl;
    std::cout << "a->index_of(4)\t\t\t" << a->index_of(4) << std::endl;
    std::cout << "a->index_of(7)\t\t\t" << a->index_of(7) << std::endl;
    std::cout << std::boolalpha;
    std::cout << "a->is_empty()\t\t\t" << a->is_empty() << std::endl;
    Array<int> *temp = new Array<int>(3);
    temp->print();
    std::cout << "->is_empty()\t\t" << temp->is_empty() << std::endl;
    std::cout << "a->is_equal([0, 0, 0])\t\t" << a->is_equal(temp) << std::endl;
    delete temp;
    temp = new Array<int>(new int[5]{5, 2, 1, 1, 4}, 5);
    std::cout << "a->is_equal(";
    temp->print();
    std::cout << ")\t" << a->is_equal(temp) << std::endl;
    delete temp;
    std::cout << "a->shuffle\t\t\t";
    a->shuffle()->println();
    std::cout << "a->sort()\t\t\t";
    a->sort()->println();
    std::cout << "a->slice(1,3)\t\t\t";
    a->slice(1, 3)->println();
}
```

```cpp
template <typename T>
void print(T A[], int n)
{
    std::cout << "[" << A[0];
    for (int i = 1; i < n; i++)
        std::cout << ", " << A[i];
    std::cout << "]";
}

void testq2()
{
    int n = 4;
    int *A = new int[n]{1, 3, 5, 7};
    std::cout << "A = ";
    print(A, n);
    std::cout << std::endl;
    for (int i = -3; i < 10; i++)
    {
        std::cout << std::boolalpha << "is_k_bit_int_not_in_A(" << i << ", A, " << n
<< ")  \t-> ";
        std::cout << is_k_bit_int_not_in_A(i, A, n) << std::endl;
    }
}

void testq3()
{
    int test[5] = {2, 3, 1, -4, -5};
    std::cout << "find_unformable(";
    print(test, 5);
    std::cout << ") -> \t";
    std::cout << find_unformable(test) << std::endl;
}

void testq4()
{
    int pivot = 2;
    int n = 5;
    int A[] = {5, 3, 2, 4, 1};
    std::cout << "rearrange_around_pivot(A,pivot)\n\tA = ";
    print(A, n);
    std::cout << ", \tpivot = " << pivot << " \t-> ";
    rearrange_around_pivot(A, pivot);
    std::cout;
    print(A, n);
    std::cout << std::endl;
    pivot = 3;
    std::cout << "\tA = ";
    print(A, n);
    std::cout << ", \tpivot = " << pivot << " \t-> ";
    rearrange_around_pivot(A, pivot);
```

```
    std::cout;
    print(A, n);
    std::cout << std::endl;
    pivot = 4;
    std::cout << "\tA = ";
    print(A, n);
    std::cout << ", \tpivot = " << pivot << " \t-> ";
    rearrange_around_pivot(A, pivot);
    std::cout;
    print(A, n);
    std::cout << std::endl;
}

void testq5()
{
    String *this_str = new String("Foo");
    String *other_str = new String("   Bar");
    std::cout << std::boolalpha;
    std::cout << "this        -> \t" << this_str << std::endl;
    std::cout << "other       -> \t" << other_str << std::endl;
    std::cout << "strlen()    -> \t" << this_str->strlen() << std::endl;
    std::cout << "strlwr()    -> \t" << this_str->strlwr() << std::endl;
    std::cout << "strlwr()    -> \t" << this_str->strupr() << std::endl;
    std::cout << "other       -> \t" << other_str << std::endl;
    std::cout << "strcat(o)   -> \t" << this_str->strcat(other_str) << std::endl;
    std::cout << "other       -> \t" << other_str << std::endl;
    std::cout << "strcpy(o)   -> \t" << this_str->strcpy(other_str) << std::endl;
    std::cout << "other       -> \t" << other_str << std::endl;
    std::cout << "o.strcpy(t) -> \t" << other_str->strcpy(this_str) << std::endl;
    std::cout << "this        -> \t" << this_str << std::endl;
    std::cout << "strcmp(o)   -> \t" << this_str->strcmp(other_str) << std::endl;
    std::cout << "strcmp(t)   -> \t" << this_str->strcmp(this_str) << std::endl;
    std::cout << "strcmp(\" \") -> \t" << this_str-
>strcmp(new String(" ")) << std::endl;
}
int main(int argc, char const *argv[])
{
    std::cout << "\n\nQUESTION 1 TEST OUTPUT" << std::endl;
    testq1();
    std::cout << "\n\nQUESTION 2 TEST OUTPUT" << std::endl;
    testq2();
    std::cout << "\n\nQUESTION 3 TEST OUTPUT" << std::endl;
    testq3();
    std::cout << "\n\nQUESTION 4 TEST OUTPUT" << std::endl;
    testq4();
    std::cout << "\n\nQUESTION 5 TEST OUTPUT" << std::endl;
    testq5();
    return 0;
}
```

```cpp
/**
 * @file Question1.cpp
 * @author Jackson York
 * @brief Question 1 on Exam 1 for Data Structures and Algorithms
 * @date 2021-02-25
 */

#ifndef Q1
#define Q1

#include <cstdlib>
#include <ctime>
#include <iostream>
/**
 * @brief
 *
 * Questions:
 *   - Array::is_empty()                    \n
 *   - Array::is_equal()                    \n
 *   - Array::is_equal()                    \n
 *   - Array::insert_at()                   \n
 *   - Array::filter()                      \n
 *   - Array::size()                        \n
 *   - Array::remove_at()                   \n
 *   - Array::remove_duplicates()           \n
 *   - Array::print(), Array::println()     \n
 *   - Array::sort()                        \n
 *   - Array::index_of()                    \n
 *   - Array::shuffle()                     \n
 *   - Array::slice()
 *
 * @tparam T
 */
template <typename T>
class Array
{
protected:
    int length;
    T *data;

public:
    /// runtime = 2
    Array(int length)
    {
        this->data = new T[length]{NULL}; //1
        this->length = length;            //1
    };
    /// runtime = 3n+4
    Array(T *arr, int n)
    {
```

```
    this->data = new T[n]{NULL};   //1
    for (size_t i = 0; i < n; i++) // 2n+2
    {
        this->data[i] = arr[i]; //n
    }
    this->length = n; //1
};
/// runtime = 3n+4
template <int n>
Array(T arr[n])
{
    this->data = new T[n]{NULL};
    for (size_t i = 0; i < n; i++)
    {
        this->data[i] = arr[i];
    }
    this->length = n;
};
/// runtime = 2
~Array()
{
    delete[] data;
    data = NULL;
};

/**
 * @note runtime = 3n+3
 *               = O(n)
 *
 * @brief is the array all NULL values
 *
 * @return whether the array is empty
 */
bool is_empty()
{
    for (size_t i = 0; i < length; i++) // 2n+2
        if (data[i] != NULL)            // n
            return false;
    return true; // 1
};

/**
 * @note runtime = 3n+4
 *               = O(n)
 *
 * @brief are all elements in each array equal?
 *
 * @param other Array to check
 * @return whether all elements in this equal all elements in other
 */
```

```
bool is_equal(Array<T> *other)
{
    if (this->length != other->length)
        return false;
    for (size_t i = 0; i < this->length; i++)
        if (this->data[i] != other->data[i])
            return false;
    return true;
};

/**
 * @note runtime = 1
 *               = O(1)
 *
 * @brief replaces the value at index with value.
 *
 * @param index at which to insert
 * @param value to insert
 * @return (void)
 */
void insert_at(int index, T value)
{
    this->data[index] = value;
};

/// runtime = 1, O(1)
T at(int index) const
{
    return this->data[index];
};

/**
 * @note runtime = 7n+n*runtime(f)+10
 *               = O(n) if f~O(1)
 *               = O(n*O(f)) else
 *
 * @brief filters elements by the result of f
 *
 * @param f test function
 * @return Array<T>* of elements where f(T value) returned true.
 */
Array<T> *filter(bool f(T value)) const
{
    T *arr = new T[length];          // 1
    int j = 0;                       // 1
    for (int i = 0; i < length; i++) //2n+2
    {
        if (f(this->data[i])) //n*runtime(f)
        {
            arr[j] = this->data[i]; // n
```

```
                j++;                    // n
            }
        }
        Array<T> *out = new Array<T>(arr, j); // 1 + (3n+4)
        return out;                           //1
    };

    /**
     * @note runtime = 1
     *               = O(1)
     *
     * @brief returns the size of the array.
     *
     * @return int size
     */
    int size() const { return length; };

    /**
     * @note runtime = 1
     *               = O(1)
     *
     * @brief sets the value at index to NULL.
     *
     * @param index value to remove.
     */
    void remove_at(int index)
    {
        this->data[index] = NULL; // 1
    };

    /**
     * @note runtime = 1.5n^2 + 4.5n + 2
     *               = O(n^2)
     *
     * @brief removes all values that are repeats after their first appearance.
     *
     */
    void remove_duplicates()
    {
        for (size_t i = 0; i < length; i++) // 2n+2
        {
            if (this-
>data[i] != NULL) // n | worst case scenario is a list with no duplicates
            {
                for (size_t j = i + 1; j < length; j++) // n*(2[(n-
1)/2]+3) | since j=i+1, the average length of the loop is (n-1)/2
                {
                    if (this->data[i] == this->data[j]) // n*(n-1)/2
                    {
```

```
                        // can happen a max of n-
1 times because data[i]!=NULL, so can only act on an element after data[0] once.
                        // This max does not matter, however, because reaching it wou
ld decrease the total runtime by an order of magnitude(data[i]!=NULL).
                        // therefore, the worst case is a list with no repeats.
                        this->data[j] = NULL; // 0
                    }
                }
            }
        }
    };

    /**
     * @note runtime = 4n + 4
     *               = O(n)
     *
     * @brief prints the array inline
     *
     */
    void print() const
    {
        if (this->length == 0) // 1
        {
            std::cout << "[]";
            return;
        }
        std::cout << "[" << this->data[0];  // 2
        for (size_t i = 1; i < length; i++) // 2(n-1) + 2
        {
            std::cout << ", " << this->data[i]; // 2n
        }
        std::cout << "]"; // 1
    };

    /**
     * @note runtime = 4n + 5
     *               = O(n)
     *
     * @brief prints the array with a newline at the end
     *
     */
    void println() const
    {
        print();
        std::cout << std::endl;
    }

private:
    /// runtime = 4nlogn + 13n - 4logn + 6, O(nlogn)
    static void quicksort(T *&array, int low_index, int high_index)
```

```
    {
        if (low_index < high_index) // 2n-1 | n-1 for each split, n for each end
        {
            int pivot_index = partition(array, low_index, high_index); // (n-
1)*(4(hi-lo) + 9) = (n - 1) * (4 log n + 9)
            quicksort(array, low_index, pivot_index);                  // n-1
            quicksort(array, pivot_index + 1, high_index);             // n-1
        }
    }
    /// runtime = 4n + 8 | n = hi - lo
    static int partition(T *&array, int low_index, int high_index)
    {
        T pivot_value = array[(high_index + low_index) / 2]; // 3
        int i = low_index - 1;                               // 2
        int j = high_index + 1;                              // 2
        while (true)
        {
            do
                i++;                          // n/2
            while (array[i] < pivot_value); // n/2
            do
                j--;                          // n/2
            while (array[j] > pivot_value); // n/2

            if (i >= j)   // n/2
                return j; // 1

            T temp = array[i];   // n/2
            array[i] = array[j]; // n/2
            array[j] = temp;     // n/2
        }
    }

public:
    /**
     * @note runtime = 4nlogn + 16n - 4logn + 12
     *               = O(nlogn)
     *
     * @brief sorts the array using the quicksort algorithm
     *
     * @return sorted array
     */
    Array<T> *sort()
    {
        Array<T> *out = new Array<T>(this->data, this->length); // 3n+5
        quicksort(out->data, 0, out-
>length - 1);               // 4nlogn + 13n - 4logn + 6, O(nlogn)
        return out;                                              // 1
    };
```

```
/**
 * @note runtime = 3n+3
 *               = O(n)
 *
 * @brief searches for a value and returns the index.
 *
 * @param value to find
 * @return int
 */
int index_of(T value)
{
    for (size_t i = 0; i < this->length; i++) // 2n+2
        if (this->data[i] == value)           // n
            return i;
    return -1; // 1
};

/**
 * @note runtime = 13n + 10
 *               = O(n)
 *
 * @brief shuffles the values in this array.
 *
 * @return Array<T>* containing the randomly shuffled values
 */
Array<T> *shuffle()
{
    srand((unsigned)time(0));                               // 2
    Array<T> *out = new Array<T>(this->data, this->length); // 3n+5
    for (int i = 0; i < out->length - 2; i++)               // 2n+2
    {
        int random = rand() % (out->length - i) + i; // 5n
        T temp = out->data[i];                       // n
        out->data[i] = out->data[random];            // n
        out->data[random] = temp;                    // n
    }
    return out; // 1
};

/**
 * @note runtime = 4n + 7
 * @brief Returns the slice of the array in the range [min, max)
 *
 * @param min lowest index to include (inclusive)
 * @param max highest index to include (exclusive)
 * @return Array<T>* containing the values of this in the range [min, max)
 */
Array<T> *slice(int min, int max)
{
    Array<T> *out = new Array<T>(max - min); // 4
```

```
        for (size_t i = min; i < max; i++)        // 2n+2
            out->data[i - min] = this->data[i];  // 2n
        return out;                               // 1
    };
};

#endif // !Q1
```

```cpp
/**
 * @file Question2.cpp
 * @author Jackson York
 * @brief Question 2 on Exam 1 for Data Structures and Algorithms
 * @date 2021-02-25
 */
#ifndef Q2
#define Q2

#include <cmath>

/**
 * @note runtime = 3 * n + 22
 *               = O(n)
 *
 * @brief Given an array of n positive odd integers, each represented with k=|`logn`|
+1 bits,
 * write an O(n)-time method for finding whether an integer is a k-
bit integer not in A.
 *
 * @param x The number that must be k-bits and not in A
 * @param A The array containing only positive odd k-bit integers
 * @param n The length of the array A, determines k by k=ceil(log2(n))+1
 */
bool is_k_bit_int_not_in_A(int x, int A[], int n)
{
    int k = ceil(log2((double)n)) + 1; // k=|`logn`|+1 bits
    if (k < 1)
        k = 1;
    // std::cout << "k = " << k;
    int num_of_bits_in_x = (int)(log2((double)x)) + 1;
    if (num_of_bits_in_x < 1)
        num_of_bits_in_x = 1;
    // std::cout << "\tb = " << num_of_bits_in_x << "\t";
    // I'm choosing to throw out negatives because the professor said in class that,
    // for this problem, (0=0, 1=1, 2=10, 3=11, ... )
    // This indicates that 2's complement notation (-2=10,-1=11, 0 = 0, 1=01, ...)
    // is invalid (and thus negative values are invalid).
    if (num_of_bits_in_x > k || x < 0)
        return false; // is not "a positive k-bit int"
    if (x % 2 == 0)
        return true; // is not odd, therefore is "a positive k-bit integer not in A"
    for (size_t i = 0; i < n; i++)
        if (x == A[i])
            return false; // is not "not in A"

    return true; // is "a positive k-bit integer not in A"
}

#endif // !Q2
```

```cpp
/**
 * @file Question3.cpp
 * @author Jackson York
 * @brief Question 3 on Exam 1 for Data Structures and Algorithms
 * @date 2021-02-25
 */

#ifndef Q3
#define Q3

#include <cmath>
/**
 * @note rumtime = 6n + 7
 *               = O(n)
 *
 * @brief This algorithm relies on the understanding that if the maximum value in an
 * array is added to itself, that is the highest value achievable by summing values i
n that array.
 *
 * @tparam n number of values in array A
 * @param A Array with arbitrary integers
 * @return int that cannot be formed from the sum of any two values in A
 */
template <int n>
int find_unformable(int (&A)[n])
{
    if (n < 1)
        return 1; // 1 cannot be formed from the sum of two non-existant numbers.
    int max_from_0 = A[0];
    for (int i = 1; i < n; i++) // iterate over all elements after A[0]
    {
        if (abs(A[i]) > abs(max_from_0))
            max_from_0 = A[i]; // set max if new max is found
    }
    return abs(max_from_0) * 2 + 1; // return 1 more than highest possible sum
}

#endif // !Q3
```

```cpp
/**
 * @file Question4.cpp
 * @author Jackson York
 * @brief Question 4 on Exam 1 for Data Structures and Algorithms
 * @date 2021-02-25
 */

#ifndef Q4
#define Q4

void recursive_call(int *A, int i, int n, int k);

/**
 * @note runtime = 8n+2
 *               = O(n)
 *
 * @brief Recursively rearranges A so that elements larger than k are to the right of
 other elements.
 *
 * @tparam n Array size
 * @param A Array to be rearranged
 * @param k value around which to pivot
 */
template <int n>
void rearrange_around_pivot(int (&A)[n], int k)
{
    recursive_call(A,0,n,k);
}
void recursive_call(int *A, int i, int n, int k)
{
    if (i == n) // every call of this function moves n and i closer together.
        return; // this stops the calls when all elements have been switched or skipp
ed.

    if (A[i] > k) // switch {n-1}th element and {i}th element
    {
        int temp = A[i];
        int n_1 = n - 1;
        A[i] = A[n_1];
        A[n_1] = temp;
        recursive_call(A, i, n_1, k); // moves n closer to i
    }
    else // skip {i}th element
    {
        recursive_call(A, i + 1, n, k); // moves i closer to n
    }
}

#endif // !Q4
```

```cpp
/**
 * @file Question5.cpp
 * @author Jackson York
 * @brief Question 5 on Exam 1 for Data Structures and Algorithms
 * @date 2021-02-25
 */

#ifndef Q5
#define Q5

#include "./Question1.cpp"
#include <iostream>

// #include <iostream>

// unsafe flag
struct unsafe
{
};

class String
{
    Array<char> *values = nullptr;

    /// runtime = 1
    String(Array<char> *values, unsafe u)
    {
        // unsafe, uses given value array without copying
        this->values = values;
    };

public:
    /// runtime = 3n+6
    template <int n>
    String(const char (&char_array)[n])
    { // template allows entry of only char[].
        this->values = new Array<char>(n - 1);
        for (size_t i = 0; i < n - 1; i++)
        {
            this->values->insert_at(i, char_array[i]);
        }
    };
    /// runtime = 8n+11
    String(Array<char> *values)
    {
        // copies all values by filtering with a condition of true.
        this->values = values->filter([](char _) -> bool { return true; });
    };
    /// runtime = 2
    ~String()
```

```
    {
        delete[] values;  // free memory
        values = nullptr; // dereferencing
    }

    /**
     * @note runtime = 2
     *               = O(1)
     *
     * @brief length of the string
     *
     * @return int the length of the string
     */
    int strlen() const { return values->size(); }

    /**
     * @note runtime = 9n + 10
     *               = O(n)
     *
     * @brief converts all uppercase characters to their lowercase equivalents
     *
     * @return String* converted string
     */
    String *strlwr() const
    {
        // setting this_len causes an order-
n speedup as size() would be called in the for loop.
        int this_len = this->values->size();
        Array<char> *lower_arr = new Array<char>(this_len);
        for (int i = 0; i < this_len; i++)
        {
            // this can save up to 3n in runtime at face value. I could use [[gnu::pu
re]] to cause
            //     this behavior at compile time, at the expense of explicitness.
            char c = this->values->at(i);
            // constant math is calculated at compile time. ('a' - 'A')=32 is purely
for readability.
            // triconditional helps with caching.
            lower_arr->insert_at(i, c + ((c >= 'A' && c <= 'Z') ? ('a' - 'A') : 0));
        }
        // using unsafe version because upper_arr is otherwise inaccessible.
        return new String(lower_arr, unsafe());
    }

    /**
     * @note runtime = 9n + 10
     *               = O(n)
     *
     * @brief converts all lower case characters to their uppercase equivalents.
     *
```

```
     * @return String* converted string
     */
    String *strupr() const
    {
        // see strlwr() for comments, these are identical outside of logic.
        int this_len = this->values->size();
        Array<char> *upper_arr = new Array<char>(this->values->size());
        for (int i = 0; i < this_len; i++)
        {
            char c = this->values->at(i);
            upper_arr->insert_at(i, c - ((c >= 'a' && c <= 'z') ? ('a' - 'A') : 0));
        }
        return new String(upper_arr, unsafe());
    }

    /**
     * @note runtime = 4m+5p + 13  | m := this_len, p := other_len
     *               = 9n + 13     | n := (4m + 5p) / 9 = (weighted) average length o
f input strings
     *               = O(n)
     *
     * @brief appends the given string to the end of this string
     *
     * @param other string to append
     * @return String* containing the concatenated string
     */
    String *strcat(String *other) const
    {
        // setting this_len, other_len causes an order-
n speedup as size() would be called in the for loop.
        int this_len = this->values->size();
        int other_len = other->values->size();
        Array<char> *concat_arr = new Array<char>(this_len + other_len);
        for (size_t i = 0; i < this_len; i++)
            concat_arr->insert_at(i, this->values->at(i));
        for (size_t i = 0; i < other_len; i++)
            concat_arr->insert_at(this_len + i, other->values->at(i));
        return new String(concat_arr, unsafe());
    }

    /**
     * @note runtime = 4n + 7
     *               = O(n)
     *
     * @brief copies this string into the destination string
     *
     * @param dest_str the destination string
     * @return bool whether the copy succeeded
     */
    bool strcpy(String *dest_str) const
```

```
    {
        int this_len = this->values->size();                    // 2
        if (this_len > dest_str->values->size())                // 2
            return false;                                       // cannot copy strin
g to other.
        for (size_t i = 0; i < this_len; i++)                   // 2n+2
            dest_str->values->insert_at(i, this->values->at(i)); // 2n
        return true;                                            // 1
    }

    /**
     * @note runtime = 10n + 13
     *               = O(n)
     *
     * @brief compares this with other, returning
     * 0 if equal,
     * 1 if greater than,
     * -1 if less than
     *
     * @param other string to compare
     * @return int ( == 0; > 1; < -1 )
     */
    int strcmp(const String *other) const
    {
        int this_len = this->values->size();
        int other_len = other->values->size();
        for (size_t i = 0; i < this_len && i < other_len; i++)
            if (this->values->at(i) < other->values->at(i))
                return -1;
            else if (this->values->at(i) > other->values->at(i))
                return 1;
        return this_len == other_len ? 0 : (this_len > other_len ? 1 : -1);
    }
    friend std::ostream &operator<<(std::ostream &o, String *s);
};

std::ostream &operator<<(std::ostream &o, String *s)
{
    o << "\"";
    for (int i = 0; i < s->strlen(); i++)
        o << s->values->at(i);
    return o << "\"";
}

#endif // !Q5
```