Efi Arazi School of Computer Science

Introduction to Computer Science

Midterm Examination Solution and Grading 2021

* The exam lasts 2 hours. There will be no time extension.
* Use your time efficiently. If you get stuck somewhere, leave the question and move on to another question.
* Use of digital devices, books, lecture notes, and anything other than this exam form is forbidden. All the materials that you need for answering this exam are supplied with the exam.
* Answer all questions on the current exam form.
* Answer all the questions on the exam pages. **Don’t write anything on the back of the pages**. Only the front pages are scanned for grading. You will get some blank pages for draft (טיוטה).
* You can answer any question in either English or Hebrew.
* If you feel a need to make an assumption, you may do so as long as the assumption is reasonable and clearly stated.
* If you can't give a complete answer, give a partial answer. A partial answer will award partial points.
* If you are asked to write code and you feel that you can't write it, you may describe what you wish to do in natural language (English or Hebrew). A good explanation will award partial credit.
* If you are asked to write code that operates on some input, there is no need to validate the input unless you are explicitly asked to do so. Likewise, if you are asked to write a function that operates on some arguments, there is no need to validate the arguments unless you are explicitly asked to do so.
* There is no need to document the code that you write, unless you want to communicate something to us.
* The code that you will write in the exam will be judged, among other things, on its conciseness, elegance, and efficiency. Unnecessarily long or cumbersome code will cause loss of points, even if it provides the correct answer.
* No points will be taken for trivial syntax errors. For example, instead of writing System.out.println(x) you can write println(x).

Good Luck!

Grading notes:

1. Correct solutions that are different than those shown in this solution should get full points, but see the next point.

2. Correct solutions that are inefficient, sloppy, or cumbersome should lose up to 1/3 of the points.

3. Don’t reduce points for syntax errors and trivial things, like using “print” instead of “System.out.print”

4. When reducing points, you must add a written comment that explains the reduction.

Consider the following program:

public class Mystery {

public static void main(String[] args) {

int x = 3;

int[] y = { 1, 2, 3 };

triple1(x);

triple2(y);

System.out.println(x);

for (int i = 0; i < x; i++) {

System.out.print(y[i] + " ");

}

}

public static void triple1(int a) {

a = 3 \* a;

}

public static void triple2(int[] a) {

for (int i = 0; i < a.length; i++) {

a[i] = 3 \* a[i];

}

}

}

1. (10 points) What will be the output of this program?

(a) Write the exact output below. (b) Explain your answer.

Output:

3

3 6 9

Explanation:

The variable x did not change, since a function cannot mutate a variable that represents a primitive value. The elements of y changed, since a function can mutate the elements of a given array.

**Grading note:** 5 points for (a), 5 points for (b).

Questions 2, 3, 4, 5 deal with the class Sets, described in the help page. Take five minutes to read this help page now.

In every one of the functions below, If the function accepts a set as a parameter, assume that the set has at least one element.

2. (10 points) Implement the following function. Your code can use any one of the Sets class functions listed in the help page, even if the function is not implemented.

/\*\* Returns the intersection of the three sets. The intersection of set1, set2, and set3

\* is the set containing all the elements that are both in set1, set2, and set3. \*/

public static int[] intersection (int[] set1, int[] set2, int[] set3) {

// Write your code here:

Answer:

public static int[] intersection (int[] set1, int[] set2, int[] set3) {

return intersection(intersection(set1, set2), set3);

}

**Grading note:** Correct and “direct” solutions that wrote loops etc.: Reduce 2 points and write “Code should be reused whenever possible. You should have used the available intersection function”.

3. (5 points) The Sets class includes two functions that have the same name (intersection).

(a) How does the Java compiler know which function to call?

(b) What is the name of this programming technique?

(c) Is it recommended to use this technique? Why or why not?

Answer: (a) The compiler knows which function to call according to the number of the arguments in the function call. (b) This technique is called *overloading*. (c) Yes, it is recommended, since it makes the code more readable.

**Grading note:** (a) 2 points, (b) 2 points, (c) 1 point.

4. (15 points) Implement the following function. Your code can use any one of the Sets class functions listed in the help page, even if the function is not implemented. Tip: This problem can be solved efficiently by going once over all the elements of one set, and doing something, and once over all the elements of the other set, and doing something. Try to find and implement such an algorithm.

/\*\* Checks if set1 dominates set2. Set1 dominates set2 is every element of set1

\* is greater than every element of set2. \*/

public static boolean dominates(int[] set1, int[] set2) {

// Write your code here:

Answer:

public static boolean dominates(int[] set1, int[] set2) {

int min = set1[0];

for (int i = 0; i < set1.length; i++) {

if (set1[i] < min) {

min = set1[i];

}

}

for (int i = 0; i < set2.length; i++) {

if (min <= set2[i]) {

return false;

}

}

return true;

}

**Grading note:** Correct solutions that are O(N2) instead of O(N): Reduce 3 points and write “Inefficient solution. See the exam solution”.

5. (20 points) Implement the following function.

/\*\* Returns a set containing the odd elements of the given set (elements that are not multiples of 2). \*/

public static int[] oddValuesOf(int[] set) {

// Write your code here:

Answer:

public static int[] oddValuesOf(int[] set) {

// Creates a temporary array whose size is the maximal possible size of the answer

int[] temp = new int[set.length];

// Adds the set's odd elements

int size = 0;

for (int i = 0; i < set.length; i++) {

if ((set[i] % 2) != 0) {

temp[size] = set[i];

size++;

}

}

// Creates and returns an array that has the right size

int[] ans = new int[size];

for (int i = 0; i < size; i++) {

ans[i] = temp[i];

}

return ans;

}

**Grading note:** Another approach is to compute in advance the length of the final array, and then build it. This solution may be less efficient, but it saves memory (sort of). Therefore, don’t reduce points for this approach.

Question 6 deals with the class Tables, described in the help page.

Take five minutes to read this help page now.

6. (20 points) Implement the following function.

/\*\* Returns a one-dimensional array containing the averages of the columns

\* of the given two-dimensional array. \*/

public static double[] colAverages(int[][] arr) {

// Write your code here:

Answer:

/\*\* Returns a one-dimensional array containing the averages of the columns

\* of the given two-dimensional array. \*/

public static double[] colAverages(int[][] arr) {

int N = arr.length; // number of rows

int M = arr[0].length; // number of columns

double[] average = new double[M];

for (int j = 0; j < M; j++) {

double sum = 0;

for (int i = 0; i < N; i++) {

sum = sum + arr[i][j];

}

average[j] = sum / N;

}

return average;

}

7. (20 points) In HW4 you had to implement a function that takes a String consisting of digit characters, like "536", and returns the int value 536. The intToString function performs the opposite operation: it takes an int value, like 536, and returns the string "536". For example, the expression intToString(73) + intToString(73) results in the string "7373".

Implement the function. If you think that you need it, you may consult the ASCII table given in the help pages. Note: A trivial solution is to simply “add” the given number to an empty string. You have to implement the real thing, which is handling each digit separately and building the string incrementally, one step at a time.

/\*\* Returns the string representation of the given integer value.

\* Assumes that the integer is greater than 0. \*/

public static String intToString(int x) {

// Write your code here:

Answer:

public static String intToString(int x) {

String ans = "";

while (x > 0) {

int rightMostDigit = x % 10;

x = x / 10;

char c = (char) (rightMostDigit + '0');

ans = c + ans;

}

return ans;

}

Another possible solution

public static String intToString(int x) {

String ans = "";

while (x > 0) {

int rightMostDigit = x % 10;

x = x / 10;

ans = rightMostDigit + ans;

}

return ans;

}

(End of Exam, Help pages follow)

The Sets class

/\*\* A library of operations on sets of int values. A set is a collection of values without repetition and

\* without order. For example, 2, 5, 2, 7 is not a set. {2, 5, 7} is a set. The order of the elements in a set is  
 \* insignificant. For example, {2, 5, 7} is the same set as {5, 7, 2}. \*/

public class Sets {

public static void main(String[] args) {

int[] s1 = { 3, 8, 5, 2 };

int[] s2 = { 2, 9, 8, 6, 5 };

int[] s3 = { 2, 8, 4 };

int[] s4 = { 5, 3 };

System.out.println(subsetOf(s4, s1));

println(intersection(s1, s2, s3));

println(oddValuesOf(s2));

}

/\*\* Checks if the value e appears in the set.

\* For example, if the set is { 7, 2, 5 } and e is 2, returns true.

\* if the set is { 7, 2, 5 } and e is 3, returns false. \*/

public static boolean elementOf(int e, int[] set) { // Code not shown }

/\*\* Checks if set1 is a subset of set2.

\* Set1 is a subset of set2 if every element of set1 is an element of set2. \*/

public static boolean subsetOf(int[] set1, int[] set2) {

// Question 2

}

/\*\* Returns the intersection of the two sets. The intersection of set1 and set2

\* is the set containing all the elements that are both in set1 and in set2. \*/

public static int[] intersection (int[] set1, int[] set2) { // Code not shown }

/\*\* Returns the intersection of the three sets. The intersection of set1, set2, and set3

\* is the set containing all the elements that are both in set1, set2, and set3. \*/

public static int[] intersection (int[] set1, int[] set2, int[] set3) {

// Question 3

}

/\*\* Returns a set containing the odd elements of the given set (elements that are not multiples of 2). \*/

public static int[] oddValuesOf(int[] set) {

// Question 5

}

/\*\* Prints the elements of the set, and then moves the cursor to the next line. \*/

public static void println(int[] set) { // Code not shown }

} // End of the Sets class

**Executing the program**

% java Sets

true

{ 8, 2 }

{ 9, 5 }

The Tables class

public class Tables {

public static void main(String args[]) {

int[][] a = { { 1, 2, 4, 0 },

{ 0, 1, 0, 2 },

{ 2, 3, 5, 1 } };

println(colAverages(a));

}

/\*\* Returns a one-dimensional array containing the averages of the columns of the given

\* table (a two-dimensional array in which all rows have the same number of elements). \*/

public static double[] colAverages(int[][] arr) {

// Question 6

}

/\*\* Prints the given one-dimensional array. \*/

public static void println(double[] arr) {

for (int i = 0; i < arr.length; i++) {

System.out.print(arr[i] + " ");

}

System.out.println();

}

}

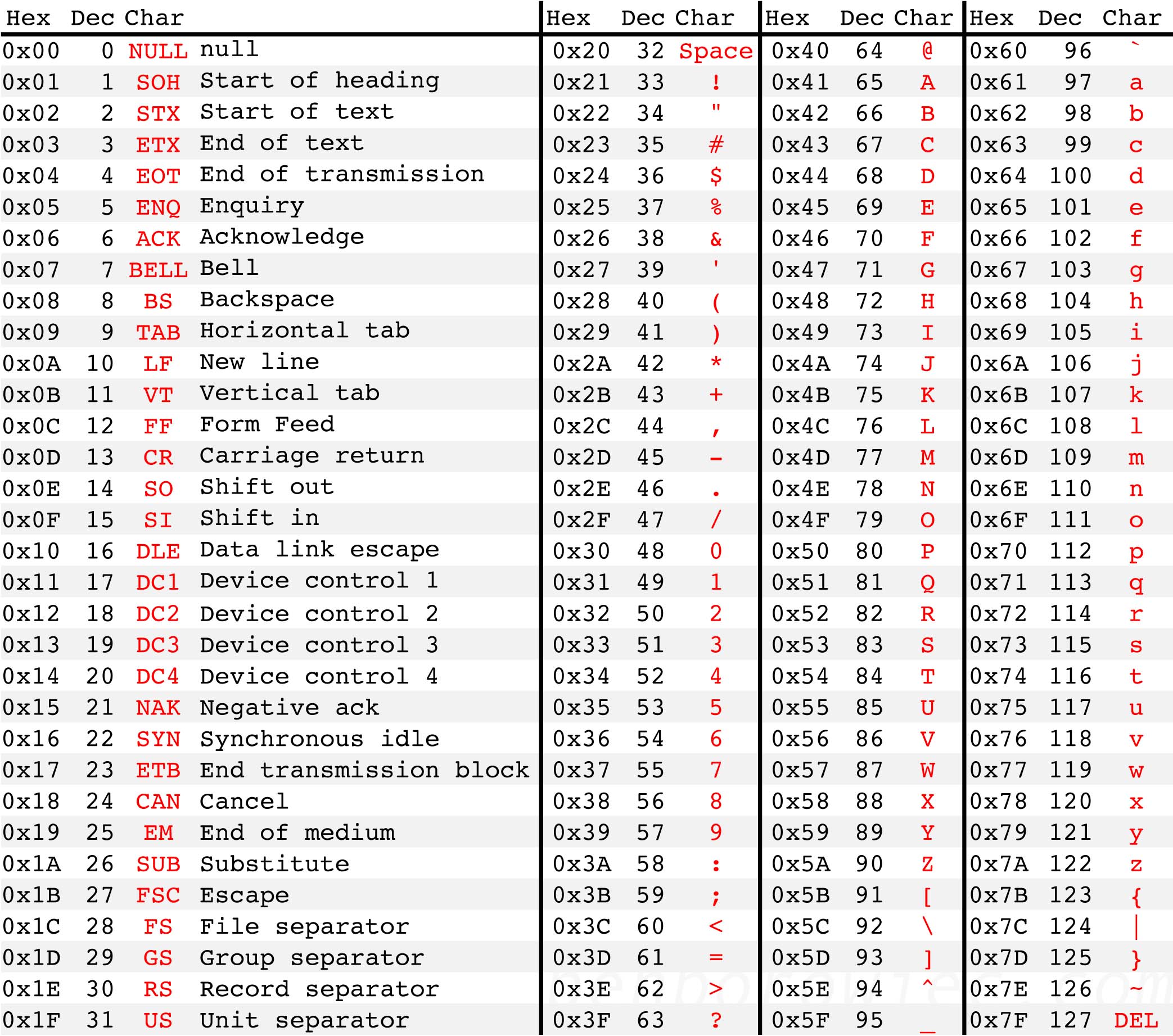
**Executing the program**

% java Tables

1.0 2.0 3.0 1.0

ASCII codes

The following table lists the character codes of a subset of commonly used characters:



Arithmetic operations

In case you will need this operation and have forgotten the syntax:

The “modulo” operation x % y computes the remainder (שארית) of the integer division x / y.

For example, 17 % 5 gives 2.