Name: Key UTA ID:

Instructions:

1. The test is worth 100 points. The point value of each question is given with the question. There are also extra credit questions at the end.
2. Read all of the instructions for each question and answer what is asked. Do not write down random stuff if you don’t know the answer.
3. All questions that have the same question number are related to each other. However, they are not all necessarily dependent on each other so you can skip around if needed.
4. The test is open book and open notes for all printed and hand-written material. You may NOT bring an electronic book or any electronic device to use during the test (no computer, no smart phone, etc.) You may use as much printed or written material as desired including copies of code examples.
5. You will write your answers on the test pages. If additional space is needed, you may use the backs or bottoms of the pages or extra paper. Please make a note on the test page whenever your answer continues onto another page and indicate where the answer is. Please staple all extra pages to your test when you turn it in.
6. Please write legibly and large enough to read easily. Your writing should be readable if the test is sitting on a desk in front of me. I am not looking for perfect handwriting but it does need to be legible. I will deduct points if your answers are much more difficult to read than those of the general student.
7. For multiple choice questions, circle the letter of your answer choice. Circle only one choice for your answer unless the instructions direct otherwise.
8. If you have a question during the test, please raise your hand. The TA and/or I will be available to come hear your question. Sometimes we may not be able to answer your question because it gives you too much information but you should always ask.
9. You have 1 hour and 20 minutes to complete the test.

1. Use the given code for **BookData.java** to answer the following questions:

1.a) Rewrite the nested if-else section (lines 56-71) for comparing earlyYear with the book#Years to use a switch instead of a nested if-else. Keep the same order of comparisons. {11 points}

deductions: incorrect/missing keywords switch , case , other syntax like ; {}

// worth 3 pts keyword, paren, test value

switch (earlyYear) // earlyYear is switch value ; deduct if not correct var

{

case 1967: select = 1; // must use constants ; deduct overall for variables

break; // need the break between cases ; deduct if not

case 1964: // case numbers 1,2,3,4 are incorrect; deduct if used

{

select = 2; // worth 2 pts per case

break;

}

case 1955: { select = 3; } // requires select assignments ; deduct if none

break;

case 2018:

{ // with or without parentheses is OK

select = 4;

}

break; // inside or outside parentheses is OK

}

1.b) What happens if the user types in a year value of “7818.1”? Describe in detail what will happen and why it happens. {6 points}

description: uncaught InputMismatchException causes it to crash (3 pts)

why? Error with the data types because 7818.1 is not an integer (3 pts)

1.c) Assuming the code for the program is completed logically following the instructions in the comments, what output should be produced if the user enters 2002 and ‘T’ as the inputs?

{4 points}

You chose TV show Firefly airing in 2002

Deductions: Wrong statement, quote marks

Partial credit for word Firefly alone

1.d) What code would you write to complete the program at the comment that says “// What kind of message would make sense here?” Give one line of Java code. {3 points}

Something like:

System.out.println("You made an invalid selection");

// message should be meaningful

Deductions: no sout, incorrect syntax, silly or inappropriate message, incomplete info in mesage

2) Match the symbol to the name. Put the correct letter in front of each symbol {1 each; 9 pts}

**C** != A. logical AND

**G** || B. end of a statement

**D** , C. not equals

**I** } D. separator

**F**  . E. less than or equals

**A** && F. dot selector

**H** > G. logical OR

**B** ; H. greater than

**E** <= I. end of a compound statement

3. Use the given code for **ThreeDigitPuzzle.java** to answer the following questions:

3.a) If the input value A is 943, then show the calculations and the resulting value of the variables below : {9 points}

hundreds = A / 100; = 943 / 100 = 9 // integer division // 1.5 pt calc; 1.5 pt answer

tens = (A % 100)/10; = (943 % 100)/10 = 43/10 = 4 // 1.5 pt calc; 1.5 pt answer

ones = A % 10; = 943 % 10 = 3 // 1.5 pt calc; 1.5 pt answer

there were problems with modulo (remainder)

3.b) The program currently asks the user to decide how many times to run the puzzle before they enter their first number. Assume instead that we want to let the user run the puzzle one time and then decide if they want to run it again. Answer the following questions with this in mind.

3.b.i) What lines would you remove so that the program would not loop at all in its given form? Give the line numbers and why you would remove each of those lines, i.e. list the line number then put why you would remove it. (You can ignore declarations and lines with curly braces only)

{6 points}

25 – Don’t need to ask the question // 1 pt per line #; 1 pt per reason

26 – No input to read from the user to loop // 1 pt per line #; 1 pt per reason

30 – No loop control needed // 1 pt per line #; 1 pt per reason

Deductions: removing line 27; if you removed the boolean, you had a point deducted because the boolean was not part of the loop control

3.b.ii) If we want to let the user decide on running the program again, what type of loop structure would you choose and why? {4 points}

while loop (or do-while loop) because we don’t know how many times it will run

// 2 pts loop type; 2 pts reason

Partial credit : for loop with meaningful reason

3.b.iii) Write the line(s) of code for the top of the loop structure you would use and give the line number(s) you would put this code on. Write the line number on the far left then write the line of Java code beside it. (You can indicate between lines with .1, .2, etc. ex*. 25.1* ) {9 points}

*Example of how to write your lines of code for this problem:*

*21.1 int temp;*

[This is example code. Yours would not have to be exactly like this. The comments tell what is looked for.]

// set up loop control variable

27.1 boolean runAgain = true; // 1 pt per line #; 2 line of code

// set up loop control ( and be true first time )

30 while (runAgain) // 1 pt per line #; 2 line of code

// change control so loop fails be default

31.1 runAgain = false; // 1 pt per line #; 2 line of code

3.b.iv) If there are other line(s) of code needed to complete the loop structure, write those line(s) and give line number(s) you would put this code on like above. {12 points}

// something like

// ask the user // 1 pt per line #; 2 line of code

80.1 System.out.println("Would you like to run the program again? Y or N ");

// get user answer // 1 pt per line #; 2 line of code

80.2 String again = input.next();

// set value for loop control

80.3 if (again.equals("Y")) // 1 pt per line #; 2 line of code

80.4 {

80.5 runAgain = true; // 1 pt per line #; 2 line of code

80.6 }

4.a) A palindrome is a word that is the same forwards and backwards. For example, the word “racecar” is a palindrome. Assume that you have a puzzle program like in part 3 but that it works with 4-digit numbers. In general, how can you check to see if any 4-digit number is a palindrome? Write a short algorithm in words to do this. Your algorithm should give a message AND set a flag variable to indicate if the number is or is not a palindrome. Write your algorithm as a list of steps, i.e. don’t write your algorithm as a paragraph. Remember, this is in WORDS, i.e. pseudocode, not in code, but it should be clear enough for someone else to write code to implement it. {10 points}

// Checking 4-digit numbers which means first and last digit the same AND

// two middle digits the same

// Find the digits for each place.

Get thousands digit by taking input number and dividing by 1000 // 1 pts

Get hundreds digit by taking input number remainder/modulus with 1000 and dividing by 100 // 2 pts

Get tens digit by taking input number modulus with 100 and dividing by 10 // 2 pts

Get ones digit by taking input number and taking remainder/modulus with 10 // 1 pt

// Test to see if it is a palindrome – pseudocode below is for comparing digits

// One way to test is to compare the digits

// Another way to test is to create the reversed number and then

// compare original number and reversed number

// To compare the digits do this:

If ones digit is the same as thousands digit AND tens and hundreds are the same // 3 pts

Print message stating number is a palindrome AND

Set flag variable true // 1 pt

Or

Print message stating number is not a palindrome AND

Set flag variable false // 1 pt

4.b) Given the pseudocode algorithm you just wrote, use the variables below to implement your algorithm in Java code to check to see if any 4-digit number is a palindrome. Use the boolean variable *palindrome* to indicate if the number is a palindrome or not. [You do not need to write code to read in the variable *input4digit*. You can assume that is done for you.] {8 points}

int input4digit; // value the user will input

int thousands, hundreds, tens, ones;

boolean palindrome;

// Assume that the user’s input value, input4digit,

// has been read in, checked and is a 4-digit number

// Start your code here. Be sure to give values to any variables you use

// 4 points to calculate the digits – needed for any approach

thousands = input4digit /1000;

hundreds = (input4digit% 1000) / 100;

tens = (input4digit % 100)/10;

ones = input4digit % 10;

if ((thousands == ones) && (hundreds == tens))

{

palindrome = true;

System.out.println("Your input is a 4-digit palindrome");

}

else

{

palindrome = false;

System.out.println("Your input is not a palindrome.");

}

// if reverse is calculated then replace if header above with these lines:

int input4digitreverse = ones \* 1000 + tens \* 100 + hundreds \* 10 + thousands; //1pt

if (input4digit == input4digitreverse) // 1 pt

// This code must match the pseudocode in previous question

4.c) Given a 4 digit puzzle instead of a 3 digit puzzle, how can we verify that the input number is specifically a 4-digit number? Write the Java code below that would check to see if the input value is 4 digits and print a message if it is not. You do not have to make the code do anything else – just print an error message. Use the variable names from part 4.b. {9 points}

// Need to check for less than 1000 and greater than 9999

// Can check in one condition

if ((A > 9999) || (A <1000)) // Each condition is 3 points; || 3 points

{

System.out.println("\nA is not a 4-digit number. "); // msg 3 pts

}

// OR separate conditions

if (A > 9999) // Each condition is 3 points; if-else if 3 pts

{

System.out.println("\nA is not a 4-digit number. "); // msg 1.5 pts

}

else if (A <1000)) // Each condition is 3 points;

{

System.out.println("\nA is not a 4-digit number. "); // msg 1.5 pts

}

Extra credit

XC1) For the original given program **ThreeDigitPuzzle**, assume that the user has a set of numbers, one per line, in a data file called “PuzzleTestValues.txt”. Assume the user wants to have the program test all of these values in a loop. Write the lines of code needed to connect the file to a Scanner named *inFile*. Use a File variable named *puzzleFile* to connect to the data file. {4 points}

File puzzleFile = new File("PuzzleTestValues.txt"); // 1 pt

Scanner inFile; // 1 pt

try // no try catch -2

{

inFile = new Scanner(puzzleFile); // 1 pt

}

catch(FileNotFoundException fnfe) // 1pt

{

System.out.println("No data file");

}

XC2) For the original given program **ThreeDigitPuzzle**, assume that the user has a set of numbers in a data file called “PuzzleTestValues.txt”. Using Scanner *inFile* just created in the previous question, write a loop header that would continue running until the program had read all the data in the file. You can assume that all the file data is valid and there will not be any errors to deal with. {2 points}

while (inFile.hasNextInt())

XC3) Which of the following is NOT a primitive data type? Circle the letter {2 points}

A) double

B) boolean

C) string

D) char

XC4) What do you like best about the classroom in the new SEIR building?

{ANY polite answer will receive 2 points.}