

Assignment xx Algorithmic Design Document

Make a copy before you begin (File -> Make a copy). Add the Assignment # above and complete the sections below BEFORE you begin to code and submit with your Assignment to D2L (File -> Download -> PDF). The sections will expand as you type.

zyBooks

Add your zyBooks screenshots for the % and assigned zyLabs completions below. Required percentages: all assigned zyLabs, Challenge Activity with at least 70%, and Participation Activity with at least 80%.

zyLabs, Challenge, and Participation % Screenshot:

3. Branches

100% 100% 100% ▾

Assigned zyLabs completion Screenshot:

3.17 LAB: Remove gray from RGB

100% ▾

3.18 LAB: Smallest number Optional

0% ▾

3.19 LAB: Interstate highway numbers

100% ▾

3.20 LAB: Exact change Optional

0% ▾

Assignment

Program description:

This program will let you play the incredible game “Rock, Paper, Scissors!” with your computer. Just type in Rock, Paper, or Scissors and you’ll be competing with the computer immediately!

Before you begin coding, **you must first plan out the logic** and think about what data you will use to test your program for correctness. All programmers plan before coding - this saves a lot of time and frustration! Use the steps below to identify the inputs and outputs, calculations, and steps needed to solve the problem.

Algorithmic design:

a. Identify all of the user input. What are the data types of the inputs? Define the input variables.

userInput as Char

b. Describe the program output. What is displayed to the user? What are the data types of the output? Define the output variables.

The program will output the following strings:

A welcome message;
Prompt for input with instructions;
- - Error if prompt does not match allowed input;
Computers randomly selected hand;
Outcome of user hand vs. computer hand;
A goodbye message;

c. What calculations do you need to do to transform inputs into outputs? List all formulas needed, if applicable. If there are no calculations needed, state there are no calculations for this algorithm.

Arithmetic:

```
srand((int)time(0)) //Random seed declaration  
randomNum = rand() % 3 //Set randomNum 0, 1, or 2
```

Comparisons:

```
userInput[0] == "r" || "R"  
userInput[0] == "p" || "P"  
userInput[0] == "s" || "S"
```

```
handUser[0] == handComp[0]  
handUser[0] == rockU[0], paperU[0], scissorsU[0]  
handComp[0] == rockU[0], paperU[0], scissorsU[0]
```

Constructors:

```
handUser[0, 1, 2, ... 7] == "Rock" || "Paper" || "Scissors"  
handComp[0, 1, 2, ... 7] == "Rock" || "Paper" || "Scissors"
```

d. Design the logic of your program using pseudocode or flowcharts. See pseudocode syntax at the bottom of this document. Here is where you would use conditionals, loops, functions or array constructs (if applicable) and list the steps in transforming inputs into outputs. Walk through your logic steps with the test data from the assignment document.

START

```
DECLARE Int randomNum  
DECLARE Char handComp  
DECLARE Char handUser  
DECLARE Char userInput  
DECLARE Char rock = "R"
```

```
DECLARE Char paper = "P"  
DECLARE Char scissors = "S"
```

```
DISPLAY "Welcome to the Rock Paper Scissors Game!"
```

```
DISPLAY "What is your play User?"
```

```
DISPLAY "Enter r, p, or s:"
```

```
INPUT userInput
```

```
IF userInput == "r" OR userInput == "R"
```

```
    SET handUser = Rock
```

```
ELSE IF userInput == "p" OR userInput == "P"
```

```
    SET handUser = Paper
```

```
ELSE IF userInput == "s" OR userInput == "S"
```

```
    SET handUser = Scissors
```

```
ELSE
```

```
    DISPLAY "Invalid play. Run the program again!"
```

```
    RETURN 0
```

```
SET seedRandom = systemTime
```

```
SET randomNum = random (0 to 2 inclusive both)
```

```
IF randomNum == 0
```

```
    SET handComp = Rock
```

```
ELSE IF randomNum == 1
```

```
    SET handComp = Paper
```

```
ELSE
```

```
    SET handComp = Scissors
```

```
DISPLAY "Computer plays"
```

```
IF handUser == handComp
```

```
    DISPLAY "It's a tie!"
```

```
ELSE IF handUser == Rock AND handComp == Paper
```

```
    DISPLAY "Computer wins!"
```

```
ELSE IF handUser == Rock AND handComp == Scissors
```

```
    DISPLAY "User wins!"
```

```
ELSE IF handUser == Paper AND handComp == Scissors
```

```
    DISPLAY "Computer wins!"
```

```
ELSE IF handUser == Paper AND handComp == Rock
```

```
    DISPLAY "User wins!"
```

```
ELSE IF handUser == Scissors AND handComp == Rock
```

```
    DISPLAY "Computer wins!"
```

```
ELSE IF handUser == Scissors AND handComp == Paper
```

```
    DISPLAY "User wins!"
```

```
ELSE
```

```
    DISPLAY "Application error. Please alert the programmer!"
```

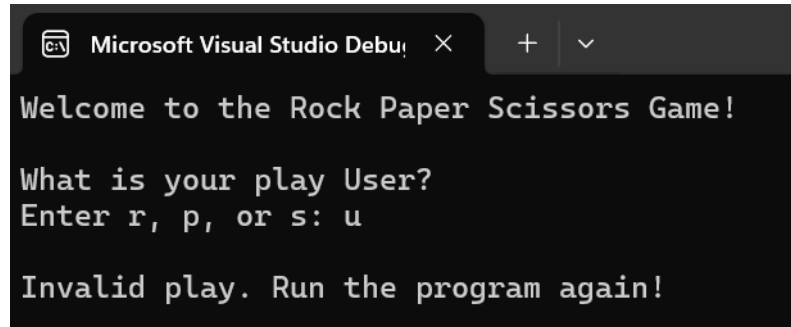
```
    RETURN 0
```

DISPLAY "Thank you for playing!"

END

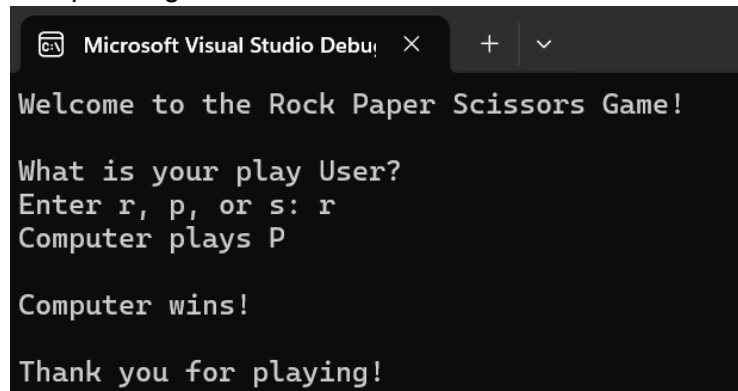
e. Include 2 Sample Program Runs for your program using your own set of data. This data set must be different from my Sample Runs in the Assignment document. This process is similar to Unit Testing and will help you test your program better.

Sample Program Run 1:



```
Microsoft Visual Studio Debug Console
Welcome to the Rock Paper Scissors Game!
What is your play User?
Enter r, p, or s: u
Invalid play. Run the program again!
```

Sample Program Run 2:



```
Microsoft Visual Studio Debug Console
Welcome to the Rock Paper Scissors Game!
What is your play User?
Enter r, p, or s: r
Computer plays P
Computer wins!
Thank you for playing!
```

Pseudocode Syntax

Think about each step in your algorithm as an action and use the verbs below:

To do this:	Use this verb:	Example:
Create a variable	DECLARE	DECLARE integer num_dogs
Print to the console window	DISPLAY	DISPLAY "Hello!"
Read input from the user into a variable	INPUT	INPUT num_dogs
Update the contents of a variable	SET	SET num_dogs = num_dogs + 1
Conditionals		

Use a single alternative conditional	IF <i>condition</i> THEN <i>statement</i> <i>statement</i> END IF	IF num_dogs > 10 THEN DISPLAY "That is a lot of dogs!" END IF
Use a dual alternative conditional	IF <i>condition</i> THEN <i>statement</i> <i>statement</i> ELSE <i>statement</i> <i>statement</i> END IF	IF num_dogs > 10 THEN DISPLAY "You have more than 10 dogs!" ELSE DISPLAY "You have ten or fewer dogs!" END IF
Use a switch/case statement	SELECT <i>variable or expression</i> CASE <i>value_1</i> : <i>statement</i> <i>statement</i> CASE <i>value_2</i> : <i>statement</i> <i>statement</i> CASE <i>value_2</i> : <i>statement</i> <i>statement</i> DEFAULT: <i>statement</i> <i>statement</i> END SELECT	SELECT num_dogs CASE 0: DISPLAY "No dogs!" CASE 1: DISPLAY "One dog.." CASE 2: DISPLAY "Two dogs.." CASE 3: DISPLAY "Three dogs.." DEFAULT: DISPLAY "Lots of dogs!" END SELECT
Loops		
Loop while a condition is true - the loop body will execute 0 or more times.	WHILE <i>condition</i> <i>statement</i> <i>statement</i> END WHILE	SET num_dogs = 1 WHILE num_dogs < 10 DISPLAY num_dogs, " dogs!" SET num_dogs = num_dogs + 1 END WHILE
Loop while a condition is true - the loop body will execute 1 or more times.	DO <i>statement</i> <i>statement</i> WHILE <i>condition</i>	SET num_dogs = 1 DO DISPLAY num_dogs, " dogs!" SET num_dogs = num_dogs + 1 WHILE num_dogs < 10
Loop a specific number of times.	FOR <i>counter</i> = <i>start</i> TO <i>end</i> <i>statement</i> <i>statement</i> END FOR	FOR count = 1 TO 10 DISPLAY num_dogs, " dogs!" END FOR
Functions		
Create a function	FUNCTION <i>return_type</i> <i>name (parameters)</i> <i>statement</i> <i>statement</i> END FUNCTION	FUNCTION Integer add(Integer num1, Integer num2) DECLARE Integer sum SET sum = num1 + num2 RETURN sum END FUNCTION
Call a function	CALL <i>function_name</i>	CALL add(2, 3)

Return data from a function	RETURN <i>value</i>	RETURN 2 + 3
-----------------------------	---------------------	--------------