Homework 1

Akash Dansinghani

Professor Umoja

Due 09/27/20

Programming Language Concepts

1.

The set of string is followed by the characters a, b, c.

S puts out abc.

A has the output abbbbbbbbbbc.

B has the output aaaaaaaaaabbcc.

C has the output of aabcc

2. Bbaab and baab. The reason why these two strings are the only one that is generated from the given options is that it states in the language that the language starts with b and since it starts with a b is ends with a b and it has an even amount of a’s in the middle of the expression. That is why bbbab is not an answer since there is only one a in the middle of the string. Bbaaaaa is not an answer since there is no b at the end of the string.

3. One of the requirements is that the loop has to be terminated else it will produce nothing. This is called a loop variant which states that the value of the loop will monotonically decrease with each time the while loop is executed until it is terminated and is not continuous. The next requirement is the axiom or the axiomatic semantics which is a precondition that says the relationships and constraints within the variables are all set to be true. This statement is like a hypothesis when proving a theory except it provides the rules of semantics to prove a program is true. A requirement as well as the weakest precondition that has to be defined and computable to get to the postcondition. Another requirement is to prove the loop invariant is true before the loop is executed. The three steps to prove this is initialization (prove each loop is true before it runs), Maintenance (the code stays true with each iteration), and termination (the end will result will produce the anticipated result). It also aids in the design of the code.

Preconditon => Invariant

// P = Precondition and I = invariant which is makes the precondition

{Invariant and Boolean} Statement(s) {Invariant}

//This rules proves that the invariant and the Boolean expression are == Post condition

(Invariant and (not Boolean)) => Q (Post Condtion)

//Invariant and (!=) not the Boolean expression equal to the post condition

The loop ends

Proof:

power = 1;

i = 1;

while (i <= n) {

power = power \* x

i = i + 1;

}

{

power = x ^ n

Revised

int power, i, n; // intialized power, i , n

power = 1;

i = 1;

while (i <= n) {

// n has a value of zero. Might need to add n++ (counter to increment since the program will

not stop)

power = power \* n; // ; use semicolon

i = i + 1;

}

{

int x;

int power = 0

power = (x ^ n); // ; use semicolon and x was never intialized

// N is not initialized

The first thing we have to deal with similar to the proof above is the invariant. The invariant here is the power \* n where in the while loop it has to be greater than 0 in order for it to be considered a power (Otherwise, it would be 1 since n^0 = 1). After the loop is executed, we see if the value does not result in one to make sure the power == > 0. Once I becomes greater than 1, the B

4.

1. Do-while in Java

// code that pertains to the program (Usually a while loop that enters do-while loop above)

do {

// body of the loop

// statement(s) that need to be executed

} while (condition);

// execute statement if true

// if false, then exit the while loop

1. C++ if then else

if (condition) {

// code

// if the condition is true, then the if statement will be executed and will exit the //if-else loop and continue to the next part of the program (unless there is a //counter).

// if the condition is not satisfied, it will go to else statement.

} else {

// code

// code will be executed and exit the loop (unless there is a counter).

}

// code after loop if it exists

If(Boolean expression = true)

Execute line 1 (first statement)

Execute line 2 (second statement)

B = Expression

5a. A for loop is a disguised while loop (just increment the index after code)

Mf(For C, Statement, s)

if( C the L,s && if VARMAP(X,s) = error)

then error

Else

S = M(B1,s)

If (M(B2,s) = error

then error

Else if (M(C,s) == 0 or undefined)

then s

Else if (M(L,s) !=0 or undefined)

S = M(B2,s)

Else M( if C then L,M(L,s)

Vx = VARMAP(vx,s)

5b. Do while

Mr(case (Do-while) <expressions> repeat do while B)

M(do(B), statement,s)

S = M(statement,s)

if M(B, s) = 0 or undefined

then error

else if M(do(E, s) = error

then error

else if M(B, s) = true

then M(do(B,statement,s)

else M(repeat statement until B), M(Statement, s))

5c. Switch

Mr (case <expression> of)

If M(B,s) = undef

Then error

If M(Left,s) = undef

OR

If Ms(Right,s) = undef

Then error

Else if Mb(B, <operator> == ‘\*’)

Then Ms(L,s) \* Ms(R,s)

Else Mb(B, <operator> == ‘+’)

Then Ms(L,s) + Ms(R,s)

8.

public class CountDigits { **4**

public static void main(String[] args) { **11**

SimpleIO.prompt("Enter an integer: "); **7**

String userInput = SimpleIO.readLine(); **9**

int number = Integer.parseInt(userInput); **10**

int numDigits = 0; **5**

while (number > 0) { **7**

number /= 10; **5**

numDigits++; **4**

} **1**

System.out.println("The number " + userInput + " has " +

numDigits + " digits");

} **1**

} **1**

9.

1. class Thermometer {

2. private int temperature //No semicolon (syntactic error)

3. public Thermometer(int degrees) {

4. temperature = degrees; 5. }

6. public Thermometer() {

7. temperature = 0.0; // need double only int (semantic error) int double temp = 0.0

8. }

9. public void makeWarmer(int degrees) {

10. temperature =+ degrees; // += instead of =+ because it will not perform the addition to degrees the other wat (Semantic Error) (Line 10)

11. }

12. public void makeCooler(int degrees) {

13. temperature -= degrees;

14. }

15. public getTemperature() { missing return type (semantic error) Line 15

16. return temperature;

17. }

18. public string tostring() { // to String, capitalize the S which turns into: toString() (semantic error) Line 18

19. return temperature + “ degrees’; // ”degrees’ have one “ + ‘. Needs “degrees” (lexical error) Line 19

20. }

21. }

10. (Uploaded Java code and document in iCollege)