HW5

Graded

Student

Adam Fenjiro

Total Points

43.5 / 53 pts

Question 1

Q1 — **14.5** / 20 pts

1.1 a — **4** / 4 pts

- ✔ **– 0 pts** Correct
- **– 1 pt** One incorrect output
- **– 2 pts** Two incorrect outputs
- **– 4 pts** More than two incorrect outputs
- **– 4 pts** No attempt

1.2 b — **6** / 8 pts

- **– 0 pts** Correct
- **– 1 pt** One incorrect stack frame
- **– 2 pts** Two incorrect stack frames
- **– 4 pts** Three incorrect stack frames
- **– 5 pts** Four incorrect stack frames
- **– 6 pts** More than four incorrect stack frames
- **– 8 pts** No attempt
- ✔ **– 2 pts** Missing variables and parameters in stack frames

1.3 c — **4.5** / 8 pts

- **– 0 pts** Correct
- **– 3 pts** Incorrect/missing symbol table stack
- ✔ **– 1.5 pts** Partially correct symbol table stack
- ✔ **– 2 pts** Incorrect/missing x86 assembly
- **– 1 pt** Incorrect offset for g in x86 assembly
- **– 2 pts** Incorrect/missing explanation on how A finds g
- **– 8 pts** No attempt
- **– 1 pt** Partially correct explanation on how A finds g

Question 2

Q2

12 / 12 pts

✔ **– 0 pts** Correct

– 1 pt Partially correct print values for static scoping

– 2 pts Incorrect/missing print value for static scoping

– 2 pts Partially correct explanation for static scoping

– 4 pts Incorrect/missing explanation for static scoping

– 1 pt Partially correct print values for dynamic scoping

– 2 pts Incorrect/missing print value for dynamic scoping

– 2 pts Partially correct explanation for dynamic scoping

– 4 pts Incorrect/missing explanation for static scoping

– 12 pts No attempt

Question 3

Q3

9 / 9 pts

✔ **– 0 pts** Correct

– 3 pts One incorrect

– 6 pts Two incorrect

– 9 pts Incorrect/No attempt

Question 4

Q4 **8** / 12 pts

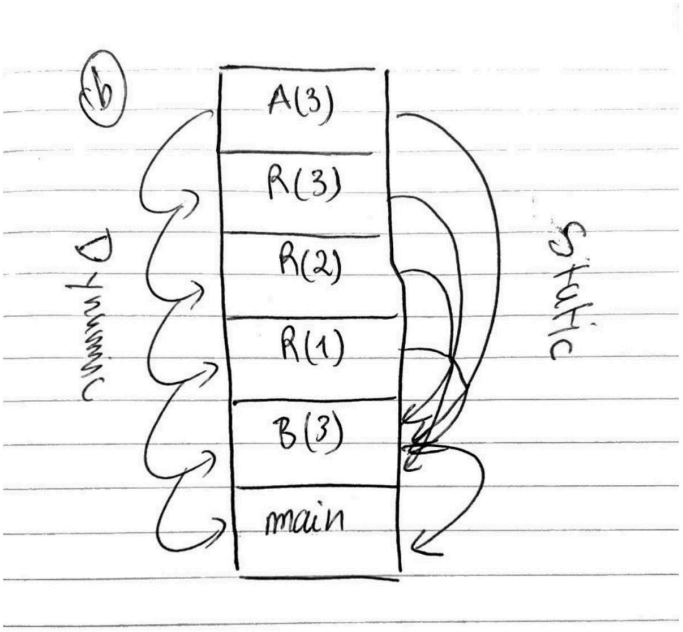
- **– 0 pts** Correct
- **– 1 pt** One incorrect in call-by-value
- **– 2 pts** Two incorrect in call-by-value
- **– 3 pts** Incorrect/missing values in call-by-value
- ✔ **– 1 pt** One incorrect in call-by-reference
- **– 2 pts** Two incorrect in call-by-reference
- **– 3 pts** Incorrect/missing values in call-by-reference
- **– 1 pt** One incorrect in call-by-value-result
- **– 2 pts** Two incorrect in call-by-value-result
- **– 3 pts** Incorrect/missing values in call-by-value-result
- **– 1 pt** One incorrect in call-by-name
- **– 2 pts** Two incorrect in call-by-name
- ✔ **– 3 pts** Incorrect/missing values in call-by-name
- **– 12 pts** No attempt

Questions assigned to the following page: 1.1, 1.2, and 1.3

Adam Fenjiro
CS4121, HW5

Problem 1:

a. The output of the program will be 9 4 2 3
b.



c. When A tries to access the variable g, I think it tries to follow the static link to the frame for B, which is the one I drew to the left. In the frame for B, it finds the variable g and assigns it the value of n.

Questions assigned to the following page: 2 and 3

Problem 2:

If the program uses Static scoping, then the output will be 1 1 2 2

This is because if the variable is not declared in the function then the global variable will be used and thus:

- s will be set to to 0, then first() will be called and set it to 1 then print x.
- print x, which is 1
- second() is called and set x to 2 then prints it x
- Print x, which is 2

If the program uses Dynamic scoping, then the output will be 1 1 2 1

This is because if the variable is not declared in the function then it will be searched in the function from where it was called, thus:

- s will be set to to 0 since the function is global scope, global x will be used
- first() will be called and set global x to 1 then print x.
- print x, which is 1.
- second() is called and set local x to 2 then prints it x.
- Print x, which will print 1 this time global x which was set to 1.

Problem 3:

(a) The program output is 3 if the language uses static scoping.
The add procedure defined in the global scope where x is initialized as 1 so when called within second then it uses global x, thus adding the global x 1 to y 2.

(b) The program output is 4 if the language uses dynamic scoping with deep binding.
The second procedure have x is redefined locally as 2, so when add is called within second then it uses the locally defined x 2 to add with y 2.

(c) The program output is 1 if the language uses dynamic scoping with shallow binding.
Procedure add is called within second so it uses the global x 1 since add was defined in the global scope. Thus adding the global x 1 to y 2.

Question assigned to the following page: 4

Problem 4:

Output of the program on call by Value:

y = 5
z = 10
x = 15

Output of the program on call by Reference:

y = 15
z = 10
x = 15

Output of the program on call by Value-Result:

y = 5
z = 10
x = 35

Output of the program on call by Name:

y = 5
z = 10
x = 15