CS 350 Assignment 5

Question 1

(a) A 1. Sub 1
2. Sub 1
3. main

B 1. Sub 1
2. Sub 1
3. Sub 1

(b) Dynamic scopic rule x = 10
Static " " x = 5

(c)
Sub 1 : a (Sub 1)
y (Sub 1)
z (Sub 1)
x (main)

Sub 2 : a (Sub 2)
b (Sub 2)
z (Sub 2)
y (Sub 1)
x (main

a

a)

1 x = 1(main)
y = 9(Sub1)
z = (Sub1)
a = 87(Sub1)

2 x = 15(Sub2)
w = 17(Sub2)
a = 13(Sub2)
y = 9(Sub1)

3 x = 15(Sub2)
b = 21(Sub3)
a = 19(Sub1)
z = 23(Sub3)
w = 17(Sub2)

4 x = 15(Sub2)
b = 21(Sub3)
a = 19(Sub1)
z = 23(Sub3)
w = 17(Sub2)

e

1 int a,b,c (def 1)
int b,c,a (def 2)

2 int a, b, c (def 1)
int b, c, a (def 2)
int c, d, e (def 3)

3 int a, b, c (def 1)
int b, c, d (def 2)
4 int a, b, c (def 1)

f

2. Page (413 - 415)

a) question 5 =>

Solution =>

a) pass by value => main program is not affected with any calculation perform on variable in sub program so for value = 2 the list will remain same
list[] = { 1, 3, 5, 7, 9 }

b) pass by reference => the change made on variables in sub program will affect the main program

swap (value, list[0]) = { 2, 3, 5, 7, 9 }

swap (list[0], list[1]) = { 3, 2, 5, 7, 9 }

swap (value, list[value]) = { 3, 2, 2, 7, 9 }

c) pass by value-result => the changes made in sub programs are sent back to main program after the function ends it will produce the same result as (b)

swap (value, list[0]) = { 2, 3, 5, 7, 9 }
swap (list[0], list[1]) = { 3, 2, 5, 7, 9 }
swap (value, list[value]) = { 3, 2, 2, 7, 9 }

b) Problem 6

Solution => static variable are global variable good when using for indexes of dynamic elements like Always bad difficult to test as they can be changed to an object easily

Where as for dynamic variables, you can directly acess them, it always indirect. The subprogram can not be history-sensitive if we use dynamic variables

c) Problem 7

Solution =>

a) pass by value => As explained earlier - pass by value does not affect the variable in main program so the list[] = {1, 3};

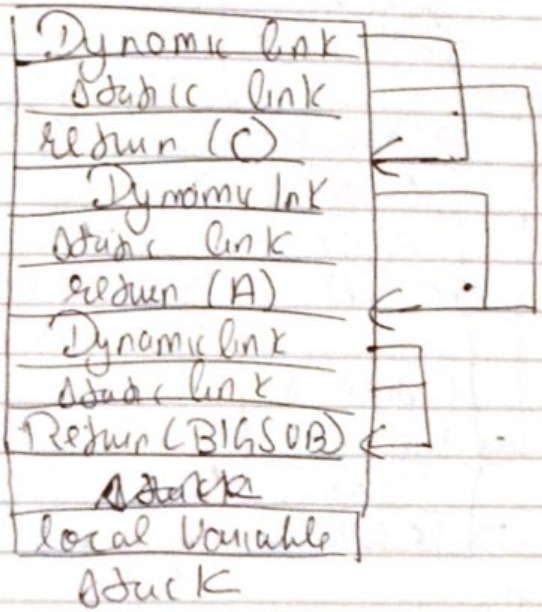
b) pass by reference => Variable are passed directly to subprogram from main and at any update or them reflects the value in main program so list[] = {2, 6};

c) pass by value result → after the function Sub program ends, these are sent back to the main program as parameters so the list [] = {2, 0}

3 Problem (442 - 445 (problem set))

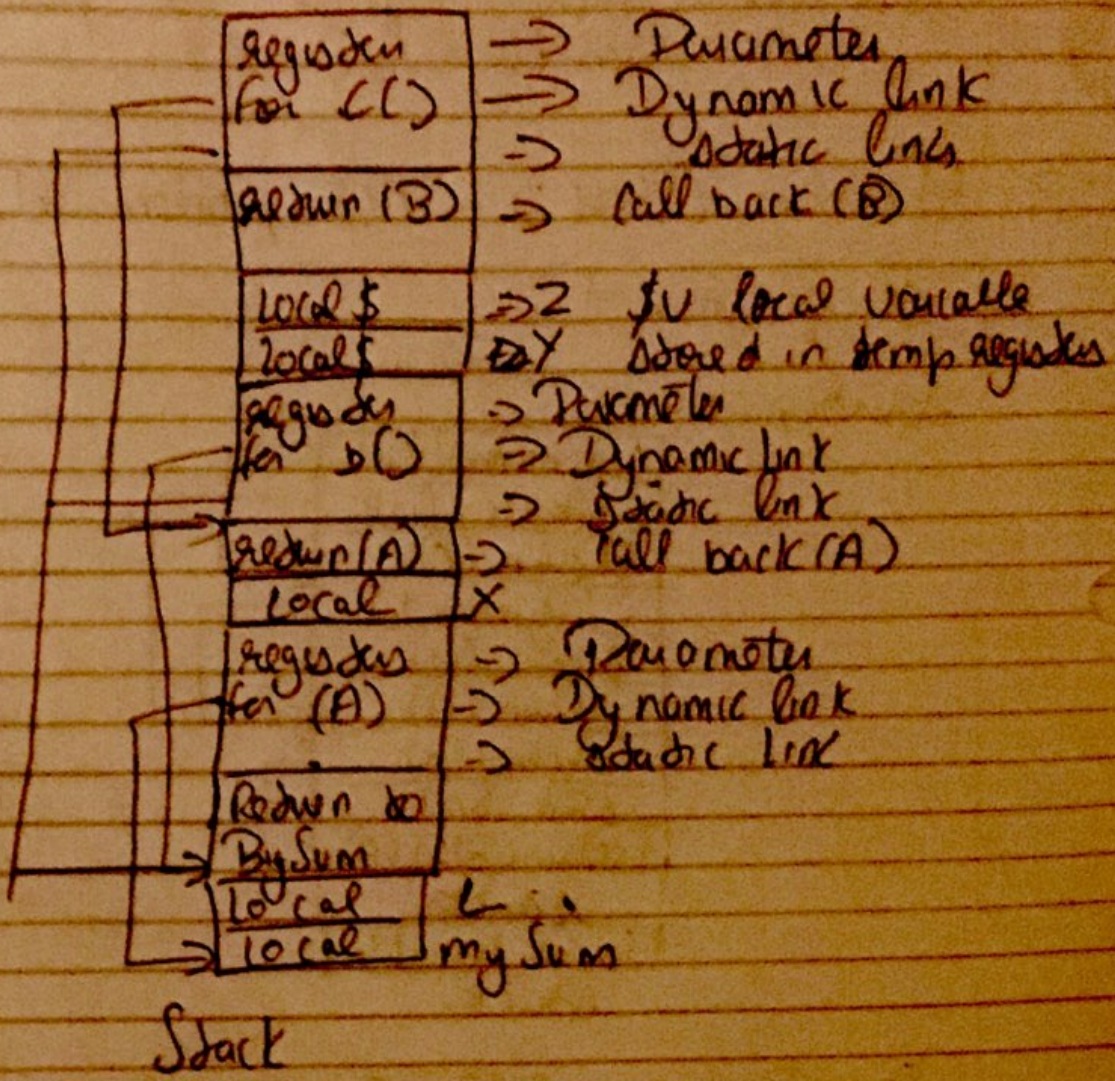
a) question 1

solution →

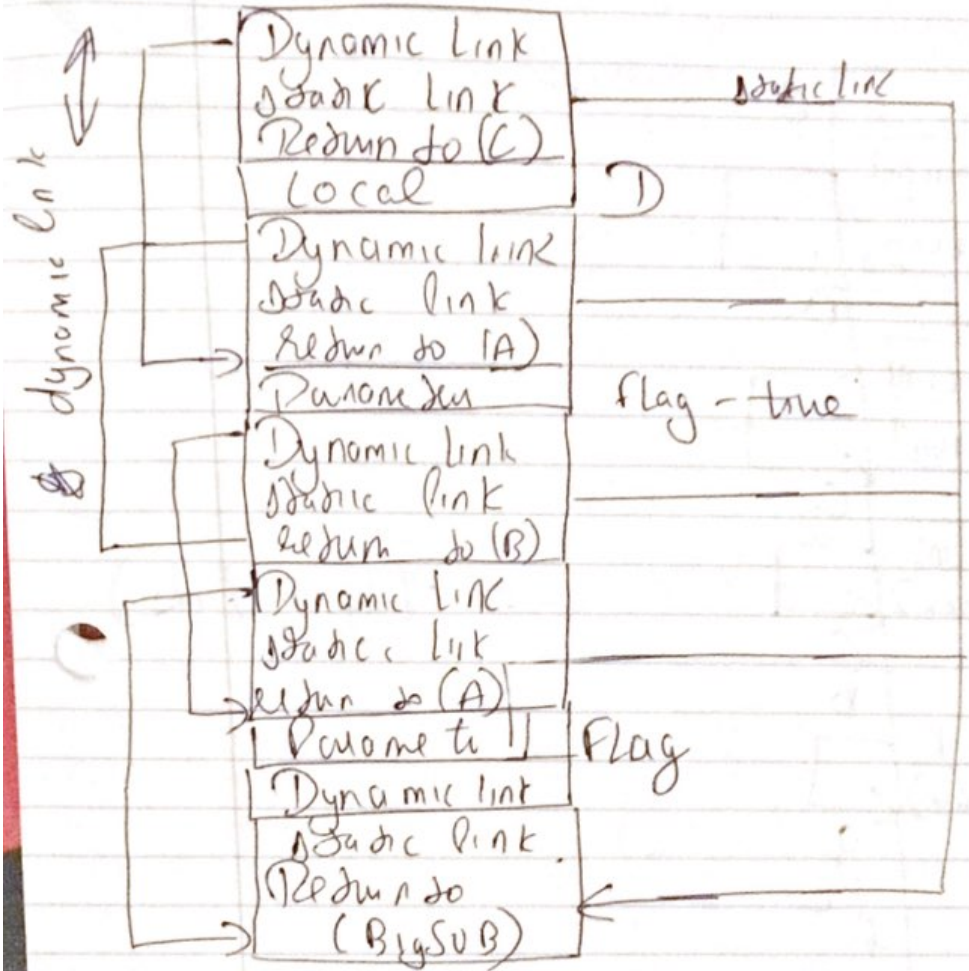


So BigSub call A the A calls C the C calls B

5) Problem 2


registers
for C()
return (B)
Local $
Local $
registers
for b()
return (A)
Local
registers
for (A)
Return to
BySum
Local
Local
Parameter
Dynamic link
Static link
Call back (B)
$V local variable
Stored in temp registers
Parameter
Dynamic link
Static link
Call back (A)
X
Parameter
Dynamic link
Static link
L
mySum
Stack

c) Problem 3


Dynamic Link
Static Link
Return to (C)
Local
D
Dynamic Link
Static Link
Return to (A)
Parameters
flag - true
Dynamic Link
Static Link
Return to (B)
Dynamic Link
Static Link
Return to (A)
Parameters
Flag
Dynamic Link
Static Link
Return to
(BigSUB)
dynamic Link

Problem 4
Solution =>

Stack

Question 5

Solution → If we use shallow access method we will use individual stack for each variable

Total variables - a, b, c, d, e, f, g

we will use 7 stacks

Stack 1 = | Func 1 / Func 1 | (a)

Stack 2 = | Func 2 | (b) Stack 3 = | Fun 2 | (c)

Stack 4 = | Fun 3 | (d) Stack 5 = | Main | (e)

Stack 6 = | Main | (f) Stack 7 = | Main | (g)

Question 7

Solution-> We encounter such similar problem when dealing with databases, for ex if we have access through a string id, the cost is high as it requires more memory, so we can for such a problem is using a Id for string which will be an integer so the cost to access using integer will be very less compared to string.

Working Principle of LISP

Lisp is a dynamic language developed in 1958 which makes it second oldest programming language. The word lisp comes from "List processor" Linked list is one of the major data structures and Lisp source code is made of lists. Lisp programs can manipulate data structure making it efficient and useful for artificial intelligence

Lisp is very fast and useful when creating a domain-specific language due to the tools like pattern matching it enable liberal use of syntactic abstraction making it very efficient. In Lisp, we can use toolchain approach to make new languages rapidly, new functions are implements as separate functions

Working Principle of Prolog

Prolog stands alongside lisp when it comes to usefulness and usability. According to one expert artificial intelligence scientist, "Prolog is one of the programming language for some basic mechanism, which ca be extemly usefull for AI programing ~~for example, it offers pattern~~

For Example Prolog offers
- pattern matching
- Automatic backtracking
- tree based data structuring mechanism

Combining these mechanisms provide a flexible framework to work with. Prolog is extensively used in expert systems for AI and is also usefull for working on medical ~~proj~~ projects.