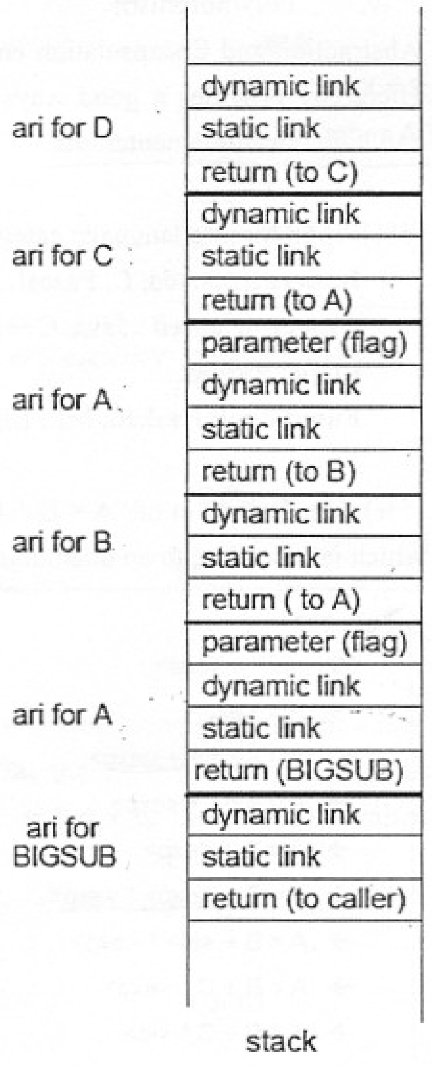
100

Final Exam for Programming Language
by Kun-Yuan Hsieh

A. (40%)Read the following sentence. Mark O if true, X if false.

O 1. (10%)Following shows the stack with all activation record instances including static and dynamic chains, when execution reaches position 1 in the following skeletal program, Assume Bigsub is at level 1.

```
procedure Bigsub
  procedure A(Flag : Boolean)
    procedure B
      ...
      A(false);
      end; --of B
    begin -- of A
    if flag
      then B;
      else C;
    ...
    end -- of A
  procedure C
    procedure D
      ... ←1
      end;
    ...
    D; -- of C
    end;
  begin -- of Bigsub
  ...
  A(true);
  ...
  end; -- of Bigsub
```



The calling sequence for this program for execution to reach D is

Bigsub calls A

A calls B

B calls A

A calls C

C calls D

X 2. (5%)In Perl,

#! /usr/bin/perl means to include perl library, like *#include "stdlib.h"* in C

/

O 3. (5%)Operator overloading means to use an operator for more than one purpose.

X 4. (5%)The four key features of Object-Oriented language are

   i. Abstraction
   ii. Encapsulation
   iii. Hierarchy
   iv. Polymorphism

   Abstraction and Encapsulation enables separating implementation and interface. Hierarchy provides a good ways for code reusing. Through polymorphism we can hide our implementations.

O 5. (5%)Programming language categories:

   Imperative: Ada, C, Pascal, …
   Object-oriented : Java, C++, Smalltalk …
   Logic: Prolog, ...
   Functional: Haskell, ML, Lisp, Scheme…

X 6. (5%) The derivation of "A = B + C * A " bellow performs a left-most derivation which is according to an unambiguous grammar.

```
<assign>
    → <id> = <exp>
    → A = <exp>
    → A = <exp> + <exp>
    → A = <id> + <exp>
    → A = B + <exp>
    → A = B + <exp> * <exp>
    → A = B + <id> * <exp>
    → A = B + C * <exp>
    → A = B + C * <id>
    → A = B + C * A
```

O 7. (5%)Following guarded statement sorts for element q1 to q4 such that q1< = q2 <= q3 <= q4

```
do q1 > q2 -> temp := q1; q1 := q2; q2 := temp;   swap(q1,q2)
[ ] q2 > q3 -> temp := q2; q2 := q3; q3 := temp;  swap(q2,q3)
[ ] q3 > q4 -> temp := q3; q3 := q4; q4 := temp;  swap(q3,q4)
od
```

B. (60%)Read select following sentence and select a correct answer.

C 1. (5%)–a) If static scope, what's X2 after execution?

a. 5

b. 10

c. 15

d. NULL

b (5%)–b) If dynamic scope, what's X2 after execution?

a. 5

b. 10

c. 15

d. NULL

```
main( ){
    int X=10;
    int X2;
    A( ){
        return X;
    }
    B( ){
        int X = 5;
        return A() + X;
    }
    X2 = B();
}
```

c 2. (10%)Assume the following rules of associativity and precedence for expressions:

Precedence:	*Highest*	***,/,not**
		+,-, &, mod
		-(unary)
		=, /=, <, <=, >=, >
	Lowest	**and**

Associativity: *Left to right*

We can show the order of evaluation of an expressions by parenthesizing all subexpressions and placing a superscript on the right parenthesis to indicate order. For example, for the expression a + b * c + d, the order of evaluation would be represented as $((a+(b*c)^1)^2+d)^3$.

What is the order the order of the expression **(a – b) / c & (d * e / a – 3)**?

a. $(((a-b)^1/c)^3 \& (((d*e)^2/a)^4-3)^5)^6$

b. $(((a-b)^6/c)^5 \& (((d*e)^4/a)^3-3)^2)^1$ (a-b)

c. $(((a-b)^1/c)^2 \& (((d*e)^3/a)^4-3)^5)^6$

d. $(((a-b)^4/c)^5 \& (((d*e)^1/a)^2-3)^3)^6$

d 3. (10%)What is the order of evaluation of problem 2, assuming that there are no precedence rules and all operators associate right to left ?

a. $((((((a-b)^1/c)^2 \& d)^3 * e)^4/a)^5-3)^6$

b. $(((a-b)^1/c)^2 \&(((d*e)^3/a)^4)^5-3)^6$

c. $((a-b)^1/(c \& (d*(e/(a-3)^2)^3)^4)^5)^6$

d. $((a-b)^5/(c \& (d*(e/(a-3)^1)^2)^3)^4)^6$

((a-b)/C&(d*e /a -3)

4. (15%)manually execute the following program

```
void main(){
        int i = 3, array[0..4] = ~~{2, 4, 6, 7, 8}~~ {2,4,6,8,10};
        swap(i, array[0]);
        swap(array[0], array[1]);
        swap(i, array[i]);
}
void swap(int a, int b){
        int temp;
        temp = a;
        a = b;
        b = temp;
}
```

for each of the following parameter-passing method, match the corresponding value of the variables i and **array** after each of the three calls to swap.

A	b	c
3, 2, 4, 6, 8, 10	2, 3, 4, 6, 8, 10	2, 3, 4, 6, 8, 10
3, 2, 4, 6, 8, 10	3, 4, 2, 6, 8, 10	3, 4, 2, 6, 8, 10
3, 2, 4, 6, 8, 10	~~6, 2, 4, 3, 8, 10~~	~~6, 2, 4, 3, 8, 10~~
	8, 2, 4, 6, 3, 10	8, 2, 4, 6, 3, 10

A 3.1 Passed by value

b 3.2 Passed by reference

b 3.3 Passed by value-result

5. (10%)Manually evaluate the following program,

C

```
( defun prog( number )
        ( cond
                ( ( <= number 0 ) 0 )
                ( ( = number 1 ) 1 )
                ( (> number 1) (* number ( prog( / number 2) ) ) )
        )
)
```

what's the result of **prog(10)**?

10 X 10 * prog(5) = 10 * 5 + prog(2)

5 X 2

a. 0

b. 30

c. 100

d. 125

|

4

6. (5%)Under UNIX, match the commands and their behaviors

a.) cd b.) ls c.) mkdir d.) cp e.) rm

C 5.1 Make a new directory

a 5.2 Change directory

e 5.3 Remove file

b 5.4 List a directory's content

d 5.5 Copy file

Bonus. Write down your suggestions to this class.

一開始我們還不是很清楚程式語言形成的概念，但是老師可能覺得這很簡單，所以講得很快，一張投影片可能只講幾十秒，然後三個小時下來連續講一兩百張，感覺滿吃不消的，尤其是 syntax analysis 的部份，大家幾乎都是到寫作業的時候才去弄懂其原理，因此寫起作業十分吃力，所以希望上課速度可以調整一下，或是刪掉一些太複雜的部份，畢竟"程式語言"應是概論性的敘述，真正要學細節可等到 compiler 或是特定程式語言課程（C, C++, JAVA..）再學。

另外作業部分希望助教多給些測值，就像 Homework 5 一樣，希望不要等到有同學問了助教回答才公佈，不然我們會改的很倉促，還會抱怨助教回答太慢，造成誤會。

不過投影片做得很詳細，比課本清楚多了，如果再加些 outline 或目錄會更好。

感謝。