CSE340 SPRING 2020
Homework 5
DUE Monday 13 April 2020

PLEASE READ THE FOLLOWING CAREFULLY

- **Your Answers for ALl problems must be types**
- **On GrAdescope, you should submit the Answers to sepArAte problems sepArAtely.**

PROBLEM 1

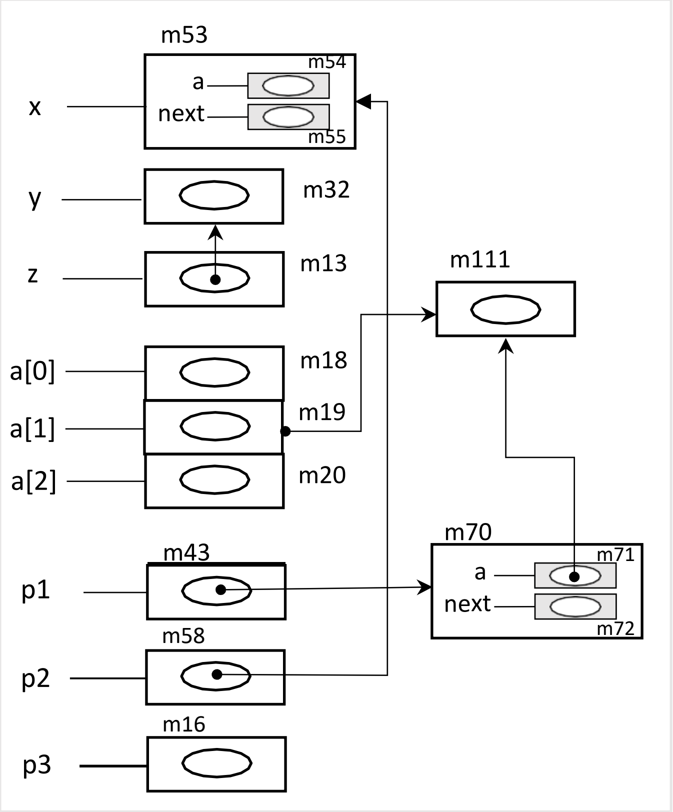
Assume stack memory allocation for nested scopes is used (which means that memory for variables in a scope is allocated on the stack and that it is deallocated when the scope is exited). Consider the following code below and the box-circle diagram to the right which illustrates the situation at **point 1.**

```
struct T {
    int *a;
    struct T* next;
};

int *y;
int **z;
int **w;
struct T   x;
struct T*  p1;
struct T*  p2;
struct T**  p3;

void f()
{ // the following malloc() call allocates m102
  // (which is not shown in the diagram because
  // f() is called after point 1)
  y = (int *) malloc(sizeof(int));
  x.a = y;
}

main()
{  p1 = (struct T*) malloc(sizeof(struct T));
   p2 = &x ;
   { int* a[3];
     a[1] = (int *) malloc(sizeof(int));
     (*p1).a = a[1];
     z = &y ;
     w = &a[1];             // point 1
   }
   f();                     // point 2
   free(y);
   (*p1).next = &x;         // point 3
   // point 4
}
```



Question 1. What is the location associated with *p1 at point 1?

m70

Question 2. What is the location associated with *z at point 2?

m32

Question 3. What is the location associated with *(x.a) at point 2?

m102

Question 4. What is the location associated with *((*p1).next) at point 2?

m72

Question 5. What are the dangling references, if any, at point 2?

None

Question 6. What are the locations that are garbage, if any, at point 3?

p1, a[1]

Question 7. What are the dangling references, if any, at point 3?

y, x.a

Question 8. Assume that the following is executed at point 3 (this applies only to this question):

```
p3 = &p1;
*p3 = &(*((*p1).next));
```

This will results in news arrows, if any, from where to where?

p3 gets a value from p1 location and *p3 gets a value from *p1.next location

Question 9. Assume the following is executed at point 4:

```
p2 = (struct T*) malloc(sizeof(struct T));
(*p1).next = p2;
p1 = p2;
```

what location become garbage due to the execution of the code?

m70

Question 10. If we execute free(p2) <u>after</u> the code above, what are the new dangling references, if any, that result from that?

p1,p2

Question 11. What is an alias of x at point 1 (the alias should have a variable other than x. Something like *&x does not count)

p2

Question 12. What is an alias of a[0] at point 1. The alias should have a variable other than a.

None

PROBLEM 2: Lambda Calculus

Question 1. Write a non-recursive lambda expression to compute the n'th Fibonacci number. The Fibannaci numbers are defined as follows

$F_1 = 1$
$F_2 = 1$
$F_n = F_{n-1} + F_{n-2}$ if $n > 1$

```
First = λT. T tru
Second = λT. (T fls(T tru))
Third = λT. (T fls(T fls))
Tuple = λc. λa. λb. pair(c pair(a b))
Init = Tuple 0 0 1
Body = λT. Tuple(plus(Second T)(Third T))(Third T)(First T)
Fib = λn. (iszero n)
      0
      (gteq n 1)
            (equal n 1)
                  1
                  (Third(prd n) (Body)( Init))
```

Question 2. Write a recursive lambda expression to compute the n'th Fibonacci number.

```
g = λFib. λn. (gteq n 2)
      (plus(Fib(prd n)) (Fib(prd(prd n))) )
      n
FibRec = fix g
```

Question 3. Write a non-recursive lambda expression to calculate the sum of the first n squares:

$1^2 + 2^2 + 3^2 + 4^2 + ... + n^2$

```
Init = pair 0 1
Body = λp. pair(plus(fst p)(times(snd p)(snd p))(succ(snd p))
Sum = λn. (iszero n)
      0
      (fst(n Body Init))
```

You should not use a closed-form formula for the sum

Question 4. Write a recursive lambda expression to calculate the sum of the first n squares:

$1^2 + 2^2 + 3^2 + 4^2 + ... + n^2$

You should not use a closed-form formula for the sum

```
g = λSum. λn. (gteq n 2)
      (plus(n prd(Sum n))
       n
SumRec = fix g
```

PROBLEM 3: Type Systems

This problem refers to the type system of the Go Programming Language. You can find the specification at https://golang.org/ref/spec. In particular you should consult the section on types: https://golang.org/ref/spec#Properties_of_types_and_values.

I expect you to consult the specification to answer the questions below.

Question 1. What is the term used in the Go language to refer to type compatibility?

Assignability

Question 2. What term is used in the Go language to refer to type equivalence?

Identity

Question 3. Can a function type with a variable number of parameter be identical to a function type with a fixed number of parameters?

No, there identity types are not the same.

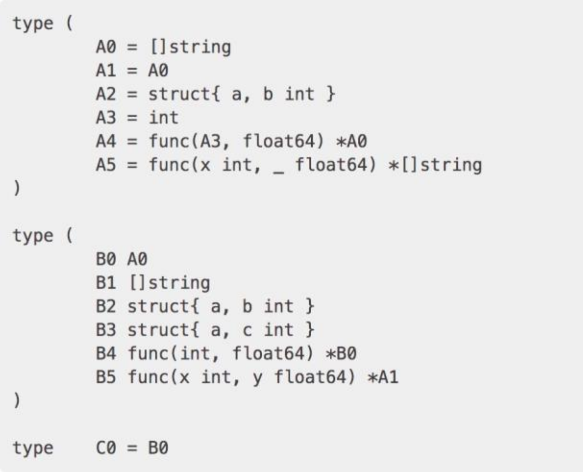
Question 4. If two types are structurally equivalent according to the definition we covered in class, would the two types be identical according to the Go language? Explain or give a counterexample.

Yes, because golang uses underlying type to determine if types are equivalent.

Question 5. If two types are identical according to the Go language, are the two types structurally equivalent according to the definition we covered in class? Explain or give a counterexample.

Yes, in order to be identical you have to have structural equivalence.

The following are examples given in the Go language spec



```
type (
        A0 = []string
        A1 = A0
        A2 = struct{ a, b int }
        A3 = int
        A4 = func(A3, float64) *A0
        A5 = func(x int, _ float64) *[]string
)

type (
        B0 A0
        B1 []string
        B2 struct{ a, b int }
        B3 struct{ a, c int }
        B4 func(int, float64) *B0
        B5 func(x int, y float64) *A1
)

type    C0 = B0
```

For the following questions, you should give explanations that are more specific than what is given in the specification document.

Question 6. Are A2 and B2 in the example above identical? Why?

Yes, because the field names and types are identical so the struct types are equivalent.

Question 7. Are struct {a, b int} and struct {a, c int} identical? Why?

No, because b and c don’t have the same field names so the struct types are not equivalent.

Question 8. Are struct {a, b int} and B2 in the example above identical? Why?

Yes, because the field names and types are identical so the struct types are equivalent.

Question 9. What is the return type of the function type A5 in the example above?

The return type is a pointer to string array

Question 10. Explain why A4 and A5 are identical.

A4 and A5 are identical because the arguments and return type are of the same type.