Programming Languages Spring 2021 Qual Exam

1. **10pts.** Give a regular expression for each of the following languages over the decimal digits {0,1,…,9}
   a. All 5-digit strings.
   b. All strings that begin with 9 and are multiples of 5.
   c. All strings that begin with 9 and contain at least one sequence of three or more consecutive 1's.

   A) [0-9] [0-9] [0-9] [0-9] [0-9]
   B) 9[0-9]*[05]
   C) 9[0-9]*111[0-9]*

2. **10 pts.** Define ambiguous as it applies to Grammars. Demonstrate that the following grammar is ambiguous.
   E -> E or E | E and E | not E | (E) | x

   Ambiguous. A sentence has two different parses from the Grammar.

   Consider x or x and x

   E -> E or E -> E or E and E ->.. x or x and x

   Vs
   E -> E and E -> E or E and E ->* x or x and X

   The first step uses a different expandsion yet same result. Ambiguous

3. **10pts** Using a functional language (like LISP, Haskell), write a function (or set of functions) which returns a list where all duplicates are removed from the original list. You may **NOT** use built in methods which perform unique directly. You may use built in methods like “member()”. For example, if the list contains 1, 1, 2, 3 and 2, then the resulting list should contain 1, 2, and 3.

```
(define uniq (f x)
   (cond ((null? x ) x)
         (( member (car x) (cdrx)) (uniq (cdr x))
         ( t ( cons (car x ) (uniq (cdr x))))))
```

4. **10pts** What are the final values of i, j, k, and *p after executing the following C program segment?

```
int i = 10, j = 20, k = 30;
int *p;
p = &k;
*p = *p + 3;
p = &j;
*p = 0;
p = &i;
*p = *p - 3
```

i=7
j= 0
k= 33
***p =7**

5. **15pts** In the following Java method, **identify all** examples of static memory allocation, stack memory allocation, and heap memory allocation (use the line numbers). Be very clear about what is where and if the allocation is explicit or implicit.

```
1 public int f() {
2   Stack<String> s = new Stack<String>();
3   String x;
4   Integer y;
5   x = "This is a string";
6   y = 1192;
7 }
```

Line 2 s is on stack, "new Stack()" is on heap (explicit)
Line 3 x is on stack
Line 4 y is on stack
Line 5 x remains on stack, points to object String ("This is a string") on heap, implicit
Line 6 y remains on stack, points to auto-box Integer on heap, implicit

6. **10pts** Explain thoroughly why objects are allocated memory from the heap and not from the stack. Discuss the challenges of objects being allocated on the heap.

The main reason why we allocate objects on the heap than on the stack is that stack elements are FIFO, they exist until the method/function activation record is removed. At this point, we would have a reference to memory which could be overwritten the next time an activation record occurs. The heap allocates memory until there are no references to it. This can be problematic is a heap object creates a circular reference chain and a garbage collection routine cannot identify that the object is not externally referenced.

7. **15pts** Implement a ***recursive*** function common_prefix in your preferred programming language. which takes two lists and returns the longest list that is a prefix of both argument lists. It should use the built-in = function to compare elements. None recursive solutions will not receive any points. Identify the language you have used for implementation. The following are some example calls:

    common_prefix([3, 4, 5], [3, 4, 6, 7]) -> [3, 4]
    common_prefix([3, 4, 5], [3, 4]) -> [3, 4]
    common_prefix([5, 3, 4], [3, 4, 5, 6, 7]) -> []

In C

```
struct node {int data; struct node * next;}

struct node * common_prefix(struct node * L1, struct node *L2)
   {   strunct node *T;

  if (L1== NULL || L2 == NULL)  return NULL;

      If (L1->data != L2 -> data ) return NULL;

     T= (struct node *) (malloc (sizeof (struct node));
     T->data = L1->data;

      T->next = common_prefix(L1->next, L2->next);

      Return T;

}
```

8. **10pts** What is "ad hoc scoping". In what context is "ad hoc scoping" used? Give an example. Why is "ad hoc scoping" important to programming language environments?

"ad hoc" scoping occurs when a function (F) is passed to another function as a parameter. The called function can then pass that function (F) to other functions. Whenever the sent function F is activated, is uses the variables and

environment that was in existence when F was first passed as a parameter. This sort of environment allows a subroutine to construct an accessor method to its environment without having to expose its variables globally. It is akin to having a helper function for those who need to know internal information to a method.

9. **10pts** Parameter passing
   Consider the following C code:

```
int i = 1;
void foo(int f, int g) {
   g = 0;
  f = f + i + 1;
}
int main( ) {
  int a[] = {1, 1, 1, 1 };
  foo(a[i+1], i);
 printf("%d %d %d %d %d \n", i, a[0], a[1], a[2], a[3]);
}
```

   a. Give the output if C uses call-by-value
   b. Give the output if C uses call-by-reference
   c. Give the output if C uses call-by-name

A) 1,1,1,1,1
B) 0, 1,1,2,1
C) 0,1,2,1,1