Paradigmas de la Programación – Recuperatorio del Tercer Parcial

19 de Junio de 2025

Apellido y Nombre: _____

Ej. 1 Ej. 2 Ej. 3 Ej. 4

1. **[10 pt.]** En el siguiente fragmento de código, identifique por su nombre y señale la línea de ocurrencia de por lo menos tres propiedades del paradigma de *scripting*.

```
$form = New-Object Windows.Forms.Form -Property @{
    StartPosition = [Windows.Forms.FormStartPosition]::CenterScreen
    Size          = New-Object Drawing.Size 243, 230
    Text          = 'Select a Date'
}

$calendar = New-Object Windows.Forms.MonthCalendar -Property @{
    ShowTodayCircle   = $false
}
$form.Controls.Add($calendar)

$result = $form.ShowDialog()

if ($result -eq [Windows.Forms.DialogResult]::OK) {
    $date = $calendar.SelectionStart
    Write-Host "Date selected: $($date.ToShortDateString())"
}
```

2. En el siguiente fragmento de código:

   a) **[5 pt.]** Identifique por lo menos tres construcciones lingüísticas con semántica concurrente y explique qué tipo de semántica concurrente están representando.

   b) **[5 pt.]** Identifique una porción del código que se podría corresponder con un monitor, también conocido como método u objeto sincronizado en algunos lenguajes.

```
void *disburse(void *arg) {

        for (i = 0; i < N_ROUNDS; i++) {
                from = rand_range(N_ACCOUNTS);
                do {
                        to = rand_range(N_ACCOUNTS);
                } while (to == from);
                pthread_mutex_lock(&accts[from].mtx);
                pthread_mutex_lock(&accts[to].mtx);
                if (accts[from].balance > 0) {
                        payment = 1 + rand_range(accts[from].balance);
                        accts[from].balance -= payment;
                        accts[to].balance   += payment;
                }
                pthread_mutex_unlock(&accts[to].mtx);
                pthread_mutex_unlock(&accts[from].mtx);
        }
```

```
18          return NULL;
19  }
20
21  int main(void) {
22          size_t i;
23          long total;
24
25          for (i = 0; i < N_THREADS; i++)
26                  pthread_create(&ts[i], NULL, disburse, NULL);
27
28          for (i = 0; i < N_THREADS; i++)
29                  pthread_join(ts[i], NULL);
30
31          for (total = 0, i = 0; i < N_ACCOUNTS; i++)
32                  total += accts[i].balance;
33
34          printf("Total money in system: %ld\n", total);
35  }
```

3. En el siguiente programa en Prolog, en los lugares del código que ocupan las letras A, y C debería haber alguna de las variables que ya se encuentran en la regla. Escriban a cuál de las variables se corresponde cada una de las letras **[5 pt.]** y expliquen la función de la variable B **[5 pt.]**

```
1  package(p1, location(a)).
2  package(p2, location(b)).
3  package(p3, location(c)).
4  package(p4, location(d)).
5
6  transport(truck, a, b).
7  transport(plane, b, c).
8  transport(train, c, d).
9  transport(ship, d, a).
10
11 connected(X, Y) :-
12      transport(_, X, Y).
13 connected(X, Y) :-
14      transport(_, Y, X).
15
16 reachable(Package, Destination) :-
17      package(A, B),
18      find_route(B, C, [Package]).
19
20 find_route(Source, Destination, Visited) :-
21      connected(Source, Destination),
22      \+ member(Destination, Visited).
23
24 find_route(Source, Destination, Visited) :-
25      connected(Source, Interim),
26      \+ member(Interim, Visited),
27      find_route(Interim, Destination, [Interim|Visited]).
```

4. **[10 pt.]** Provea por lo menos dos razones por las cuales los asistentes de programación basados en modelos de lenguaje suelen producir código con menos errores si el código forma parte de algún framework, en comparación con código situado en un programa sin inversión de control.