Paradigmas de la Programación – Recuperatorio del Tercer Parcial

18 de Junio de 2024

Apellido y Nombre: _____

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

1. **[10 pt.]** En el siguiente fragmento de código, identifique y nombre 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'
    Topmost       = $true
}

$calendar = New-Object Windows.Forms.MonthCalendar -Property @{
    ShowTodayCircle   = $false
    MaxSelectionCount = 1
}
$form.Controls.Add($calendar)

$result = $form.ShowDialog()

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

2. **[10 pt.]** Dada la siguiente base de conocimiento:

```
pension(X, pension_invalidez):- invalidez(X).
pension(X, jubilacion_mayor):- mayor_65(X), aportes(X).
pension(X, pension_mayor):- mayor_65(X).
pension(_, nada).

invalidez(fred).
mayor_65(fred).
mayor_65(joe).
mayor_65(jim).
aportes(fred).
aportes(joe).
```

   Determine qué pensiones le corresponden a fred, cuáles a joe y cuáles a teo.

3. **[10 pt.]** La palabra clave *Maybe* en Elm puede servir para dar una respuesta para algunos valores, pero no para otros.

   Explique cómo el uso de *Maybe* en el siguiente fragmento de código en Elm puede considerarse programación defensiva:

```
type alias Persona =
    { name : String
    , age : Maybe Int
    }

tom = { name = "Juan", age = Just 42 }
sue = { name = "Maria", age = Nothing }
```

4. **[10 pt.]** Identifique en el siguiente fragmento de código construcciones lingüísticas con semántica concurrente y explique qué tipo de semántica concurrente están representando.

```
void *disburse(void *arg) {
        size_t i, from, to;
        long payment;

        (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);
        }
        return NULL;
}

int main(void) {
        size_t i;
        long total;
        pthread_t ts[N_THREADS];

        srand(time(NULL));

        for (i = 0; i < N_ACCOUNTS; i++)
                accts[i] = (struct account)
                        {100, PTHREAD_MUTEX_INITIALIZER};

        for (i = 0; i < N_THREADS; i++)
                pthread_create(&ts[i], NULL, disburse, NULL);

        for (i = 0; i < N_THREADS; i++)
                pthread_join(ts[i], NULL);

        for (total = 0, i = 0; i < N_ACCOUNTS; i++)
                total += accts[i].balance;

        printf("Total money in system: %ld\n", total);
}
```