Universidad de Costa Rica

Escuela de Ciencias de la Computación e Informática.

CI0202- Principios de Informática. G-09

Examen final

I-2020

Prof. Jose Pablo Ramírez Méndez

Fecha de aplicación: Miércoles 8 de julio ¶

Inicio: 7:00 am

Fin: 9:00 am

Instrucciones:

1. El examen se compone de una parte programada con un valor de 100% para el total del 30% de este rubro en la nota final del curso. Realice su examen de manera completamente individual.
2. En cada punto se evaluará la correctitud, completitud, y buenas prácticas de programación. Estas incluyen, pero no se limitan a identificadores significativos para variables e identación adecuada. Además, si considera que requiere de más funciones o clases para dar solución a los puntos anteriores, puede implementarlas.

Uso de Drones en desastres naturales. Valor 100 pts

Enunciado

Una de las áreas más importantes de investigación y aplicación de RPA (Robotic Process Automation) es en la generación de software que minimize la intervención del ser humano en diversos procesos. El uso de drones para antender emergencias y desastres naturales representa una poderosa herramienta y campo de estudio de RPA ya que:

1. Facilitan el acceso a lugares que las personas no pueden.
2. Se mueven con gran rapidez y agilidad.
3. En general poseen sistemas de geolocalización bastantes precisos.
4. Pueden ser equipados de diversos sensores, como por ejemplo de temperatura.

La Comisión Nacional de Emergencia (CNE) planea realizar una gran inversión este año. Se planea comprar drones para atender las emergencias de inundaciones que se presentarán durante la época lluviosa. Debido a que la inversión en la compra de estos drones es muy alta, como en muchos casos, es más barato realizar simulaciones y evaluar su funcionamiento para tomar decisiones.

Cuando las lluvias son muy fuertes en las grandes llanuras de nuestro país, en muchas ocaciones personas quedan atrapadas en sus casas o automóviles y es muy difícil ubicarlas sin poner en riesgo a los rescatistas. Incluso para los heliópteros no es tarea fácil la de maniobrar sobre las zonas inundadas debido a su boscosidad.

Usted ha sido contradado/a para desarrollar un programa que permita realizar una pequeña simulación del sistema de drones que atendería este tipo de emergencias. Los drones de la simulación cuentan con las siguientes características:

1. Un número de serie único entero que es incremental, es decir, 1,2,3,4,..,n
2. Un sistema de coordenas básico basado en tres dimensiones x,y,z para conocer su posición en cualquier momento.
3. Un sistema para sensar temperatura de manera muy precisa.
4. Una memoria interna que permite almacenar la siguiente informacion: **posición de objetivos encontrados** (x,y,z) y la **temperatura** registrada para ese objetivo.

El objetivo del programa es correr una pequeña simulación en donde se crearán un conjunto de drones y se colocarán un conjunto de objetivos en la simulación. Los objetivos serán ubicados por los drones de acuerdo a su posición actual y a su distancia del objetivo.

El proceso que realiza su programa se compone de tres fases:

1. Diseño para la corrida de la simulación.
2. Diseño del controlador de la simulación.
3. Ejecución y reporte de datos.

Fase 1: Diseño para la corrida de la simulación. Valor 40 pts.

Su programa realizará la lectura de los datos de los reportes desde un **primer archivo** que el profesor le proporcionará junto con este enunciado. El archivo cumple con el siguiente formato:

```
 10.0,20.0,0.0
40.0,20.0,0.0
30.0,10.0,0.0
30.0,10.0,0.0
10.0,20.0,0.0
40.0,20.0,0.0
40.0,20.0,0.0
```

Cada línea corresponde a la información de un dron. En el archivo encontrará la posición actual del dron, es decir, sus coordenadas iniciales de la forma x,y,z separados por **coma**. Conociendo que el número de serie es incremental, las primeras dos filas del archivo representan la siguiente información:

```
 Número de serie: 1
Coordenadas: (10.0, 20.0, 0.0)

 Número de serie: 2
Coordenadas: (40.0,20.0,0.0)
```

Seguidamente, en el **segundo archivo** se almacenan de los objetivos colocadas en la simulación. Se presentan con el siguiente formato (datos separados por **punto y coma**):

```
 19;45.0;56.8;15.7
30;29.8;44.5;20.3
31;36.9;22.3;4.5
37;99.0;12.4;4.2
25;76.3;14.5;0.6
13;47.0;12.3;12.3
36;44.3;29.6;2.5
36;23.4;17.2;7.3
19;12.4;76.3;2.3
20;53.4;97.3;1.6
8;12.3;45.6;0.9
40;11.3;23.1;3.7
23;11.2;13.1;0.7
```

Por ejemplo, la primera línea contiene la siguiente información:

```
 Temperatura del objetivo: 19°C
Ubicación del objetivo: ( 29.8, 44.5, 20.3)
```

Tal vez se pregunten ¿Cómo vamos a manipular esta información? La respuesta se la pueden imaginar, sí, mediante objetos. Vamos a definir las siguientes clases para contener esta información:

Primera clase: `RegistroPersona` (15 pts)

Atributos: Implemente al menos estos atributos.

1. `posicion` : Para almacenar la posición en donde se encontró a una persona. Es una lista de con tres flotantes (x,y,z)
2. `temperatura` : Para almacenar la temperatura que registró un dron para esta persona.
3. `dron` : Para almacenar el número de serie del dron que ubicó este objetivo.

Funciones*: Implemente al menos estas funciones

1. `__init__( self, x, y, z, temp )` : constructor de la clase para inicializar atributos.

Segunda clase: `Dron` (25 pts)

Atributos: Implemente al menos los siguientes atributos.

1. `numeroSerie` : Para almacenar el número de serie único
2. `posicionInicial` : Para almacenar las coordenadas iniciales. Es una lista con tres flotantes (x,y,z)

Funciones: Implemente al menos las siguientes funciones:

1. `__init__(self, numSerie, x, y, z)` : Para inicializar los atributos de la clase.

Fase 2: Diseño del controlador de la simulación. Valor 40 pts.

Una vez que definimos las clases anteriores, es momento de diseñar como vamos a manipular esta información. Vamos a diseñar una clase `Controlador` que nos permitirá obtener los datos de los archivos y correr la simmulación. Con esta clase vamos a desempeñar las siguientes tareas:

1. Leer la información del primer archivo la información de los drones y su posición. Crear los drones y almacenarlos.

1. Leer la información del segundo archivo con los registros de personas (objetivos) junto con la temperatura y las coordenadas. Crear los registros y almacenarlos

1. Para recorrer cada dron, buscar el registro de personas más cercano y asignar el número de dron para este registro.

1. Mostrar para esta corrida, cada registro de personas y el número de dron que se le fue asignado.

Veamos como se vería la clase:

Clase `Controlador`

Atributos: Implemente al menos los siguientes atributos.

1. `listaDrones` : Para almacenar los drones de la simulación
2. `listaRegistroPersonas` : Para almacenar los registros de personas ubicados en la simulación.

Funciones: Implemente al menos las siguientes funciones:

1. `__init__(self)` : Para inicializar los atributos de la clase.

1. `cargarDrones( self, archivoDrones )` : Esta función lee las lineas del `archivoDrones` , crea las instancias de la clase `Dron` y las almacena en el atributo `listaDrones` de la clase. (10 pts)

1. `cargarRegistrosPersonas( self, archivoRegistros )` : Esta función lee las lineas del `archivoRegistros` , crea las instancias de la clase `RegistroPersona` y las almacena en el atributo `listaRegistroPersonas` de la clase. (10 pts)

1. `correrSimulacion(self)` : Ejecuta la corrida de la simulación. Por cada dron en `listaDrones` , se busca en la lista `listaRegistroPersonas` el registro con más cercano a la posición actual del dron y se el asigna el número de serie del dron actual en el atributo dron del registro. Note que si el número de drones es menor al número de registros, algunos registros quedarán sin un número de serie de dron asignado (15 pts)

Vea el anexo de cómo podría apoyarse para este cálculo

1. `mostrarResultados(self)` : Mostrar cada registro de personas almacenado en `listaRegistroPersonas` con el número de dron al que se le fue asignado. (5pts)

Fase 3: Ejecución y reporte de datos. Valor 20 pts.

Finalmente cree un pequeño programa que utilice la fase 1 y fase 2 para probar la corrida de la simulación. Su programa debe verse más o menos así:

```
controlador = Controlador()

archivoDrones = open( "primer_archivo.csv", "r" )

controlador.cargarDrones( archivoDrones )

archivoRegs = open( "segundo_archivo.csv", "r" )

controlador.cargarRegistrosPersonas( archivoRegs )

controlador.correrSimulacion()

controlador.mostrarResultados()
```

Puntos extra

Tal y como lo hemos conversado, usted tendrá la oportunidad de ganar puntos extra en este examen final si decide implementar las siguientes funcionalidades de los temas avanzados de programación:

1. Utilizar mensajes del tema avanzado 1: Interfaces Gráficas de Usuario para mostrar los resultados de invocar el método `mostrarResultados` de la clase `Controlador` . (5 pts).

1. Utilizar la interfaz gráfica para abrir archivos (con filtro para extensión csv) del tema avanzado 1: Interfaces Gráficas de Usuario, con el fin de que el usuario pueda elegir los archivos del enunciado. (5 pts)

1. Crear una de la siguientes gráficas (7 pts):

   a. Gráfica de la relación de temperatura de objetivo vs distancia con el dron más cercano. (Distancia 0 si no fue posible asignarle uno)

Anexo

Utilice el método pitagórico para encontrar la distancia entre dos vectores. Sean X y Y dos vectores de R^n:

$$D = \sqrt{\sum_{i=0}^{n-1} (X_i - Y_i)^2}$$

In [6]:
```
def distancia( vector1, vector2 ):
    return sqrt( ( vector1[0] - vector2[0] )**2 + ( vector1[1] - vector2[1] )**2 )
```

In [7]:
```
lista_vectores = []
```

In [9]:
```
from math import sqrt, inf

## Inicialicemos algunos vectores de R2 ecomo lista de dos elementos

registro_1 = [ 12, 4 ]
registro_2 = [ 7, 3 ]
registro_3 = [ 6, 4 ]

## gregar al contenedor

lista_vectores.append( registro_1 )
lista_vectores.append( registro_2 )
lista_vectores.append( registro_3 )

# Definimos una distancia inicial para encontrar la distancia más larga

vector_inicial = [ 0, 0 ]

# Si hay al menos un vector
if ( len( lista_vectores ) > 0 ):
    # Se define una distancia incial de infinito
    distancia_minima = inf
    # Tomamos cualquier vector como el mas cercano como valor inicial
    vector_mas_cercano = lista_vectores[ 0 ]
    # Iteramos por cada vector del contenedor
    for indice_vec in range( 0, len( lista_vectores ) ):
        # verificamos si la distancia al actual vector es menos a la menor act
ual
        distancia_local = distancia( lista_vectores[ indice_vec ], vector_inic
ial )
        if ( distancia_local < distancia_minima ):
            # Hacer la distancia menor como la distancia local
            distancia_minima = distancia_local
            # Hacer el vector más cercano como el actual
            vector_mas_cercano = lista_vectores[ indice_vec ]
    # Mostramos el mensaje de información
    print( "En la lista de vectores, la distancia minima es de %.2f ul hacia l
as coordenadas %s" % ( distancia_minima, vector_mas_cercano ) )
```

```
En la lista de vectores, la distancia minima es de 7.21 ul hacia las coordena
das [6, 4]
```

In [ ]: