**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**
**INGENIERÍA DE SOFTWARE II**
**SEGUNDA EVALUACIÓN - I TÉRMINO 2014**

***Nombre:*** _____ ***Matrícula:*** _____

***Sección A***

1. Las preguntas a continuación están relacionadas a refactoring:
   *a.* Refactoring es …….…… código existente alterando su…….. sin cambiar su…… ***[3%]***
   b. ¿Cuál de las siguientes actividades son refactoring? ***[2%]***
      i. Añadir nuevas funcionalidades
      ii. Corregir errores
      iii. Optimizar desempeño
      iv. Parchar vulnerabilidades de seguridad
   c. Liste al menos dos "*bad code smells*" ***[2%]***
2. Las pruebas de integración en pocas palabras: ***[5%]***
   a. prueba los componentes individuales que han sido desarrollados.
   b. prueba interacciones entre módulos o subsistemas.
   c. solo usa componentes que forman parte del sistema.
   d. prueba interfaces a otros sistemas.
3. Las pruebas de regresión deben ser desarrolladas: ***[5%]***
      v. cada semana.
      w. luego que el software ha cambiado.
      x. tan pronto como sea posible.
      y. cuando el entorno ha cambiado.
      z. cuando lo diga el director de proyecto.
   a. v & w son verdaderas, x, y & z son falsas.
   b. w, x & y son verdaderas, v & z son falsas.
   c. w & y son verdaderas, v, x & z son falsas.
   d. w es verdadera, x, v, y & z son falsas.
4. El análisis *data flow* estudia: ***[5%]***
   a. Posibles cuellos de botella en un programa.
   b. El porcentaje de cambio de valores de datos mientras un programa se ejecuta.
   c. El uso de datos en caminos a través del código.
   d. La complejidad intrínseca del código.
5. ¿Qué es un acuerdo de niveles de servicio? Describa brevemente al menos cuatro de sus componentes. ***[10%]***
6. Identifique las actividades del proceso de gestión de incidentes. Liste tres productos para gestión de incidentes. ***[8%]***

***Sección B***

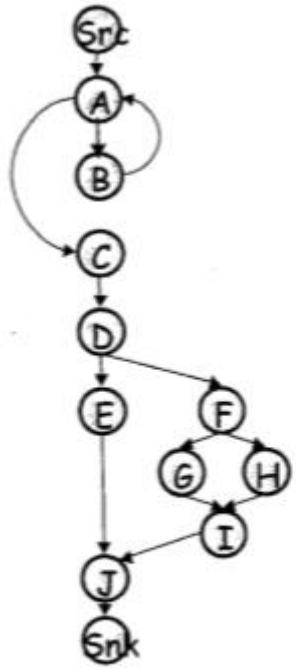
7. La función *Grading*, en la figura de la parte inferior, determina la clasificación del estudiante, dado el porcentaje de puntuación.
    a. Elabore el *Control Flow Graph* (*CFG*) para la función *Grading* ***[ 8%]***
    b. Elabore el *DD-Path Graph* para la función *Grading*. Utilizando anotaciones indique los tipos básicos de nodo que aparecen. Liste todos los paths independientes que aparecen en el grafo. ***[10%]***
    c. Especifique el número de casos de prueba necesarios para alcanzar 100% *statement coverage* ($C_0$ metric). ***[ 4%]***
    d. Especifique los casos de prueba que aseguran 100% *path coverage* ($C_1$) de la función *Grading*. Use un formulario sencillo para especificación de casos de prueba, considere una tabla con las siguientes cabeceras: ***[14%]***

| *Caso de prueba ID* | *Valores de entrada* | *Resultados esperados* |
|---|---|---|

    Como parte de su respuesta:
        - Explique qué se entiende por Path Coverage
        - Explique en detalle qué principios haya empleado para generar sus casos de prueba.
    e. ¿Ejecutar las pruebas generadas usando el método de *Path Testing* lo hace sentir más seguro que encontrará los defectos más importantes? ¿Por qué? ***[ 6%]***

```
1       function Grading
2               int score
3               string grade
4               input (score)
5               if score > 79 then
6                       result = “Excellent”
7               else
8                       if score>59 then
9                               result = “Very Good”
10                      else
11                              if score>39 then
12                                      result = “Good”
13                              else
14                                      result = “Bad”
15                              endif
16                      endif
17              endif
18              return (result)
19      end Grading
```

8. Defina la métrica ciclomática de McCabe y explique qué mide. Calcule la métrica para el Control Flow Graph dado a continuación: ***[6%]***




9. ¿Puede usted aplicar refactoring al siguiente código fuente? No hay necesidad de reescribir el código, solo marcarlo o subrayar las líneas sobre las que hará refactoring y nombrar la técnica de refactoring que usted usará: ***[12%]***

```
public int getScore()
{
        int result;
        result = (int) (Math.random()*6)+1;
        dice[0].setFaceValue(result);
        result = (int) (Math.random()*6)+1;
        dice[1]. setFaceValue(result);
        int score = dice[0].getFaceValue()+dice[1].getFaceValue();
        return score;
}
```