,d sen	e Mécnico superior de le
Departamento Electró Informática I	

APELLIDOS			
NOMBRE		Nº Ma	at.
			Calificación
ASIGNATURA:	SISTEMAS INFORMA	ÁTICOS INDUSTRIALES	
CURSO 4º	GRUPO	Diciembre 2013	

1. Problema de Análisis y Diseño Orientado a Objetos (10 puntos - 30 minutos)

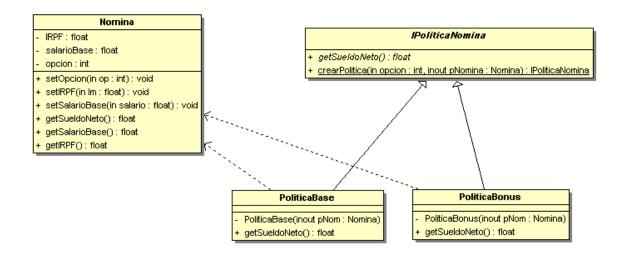
Se desea hacer una aplicación que sirva para calcular las nóminas de una compañía. Al salario base de cada empleado hay que quitarle la retención del IRPF (p. e. 20%) para calcular su salario neto. Como existen diferentes políticas salariales en la empresa, se desea hacer un programa fácilmente extensible a nuevas políticas. De momento se pretende abordar dos de ellas:

- 1. El sueldo ordinario
- 2. El sueldo con bonus, consistente en aumentar el salario base (antes de la retención) un 35%.

Se ha desarrollado el siguiente programa principal. Se pide:

- 1. Diagrama de Clases de Diseño de una arquitectura que permita una fácil extensión a nuevas políticas. Indicar los patrones utilizados. (5 puntos)
- 2. Implementación en C++ de la solución. (5 puntos)
- int main() Nomina nomina; int opcion; cout<<"1. Nomina ordinaria"<<endl; cout<<"2. Nomina con bonus"<<endl;</pre> cin>>opcion; nomina.setOpcion(opcion); cout<<"IRPF en %: "; float IRPF; cin>>IRPF; nomina.setIRPF(IRPF); cout << "Salario base: "; float salario; cin>>salario; nomina.setSalarioBase(salario); float total=nomina.getSueldoNeto(); cout<<"El salario neto es: "<<total<<endl;</pre> return 0;

1. El patrón es la Estrategia GoF. Se ha acompañado con un método de fabricación para crear la política más adecuada.



```
class Nomina;
class IPoliticaNomina{
public:
      virtual float getSueldoNeto() = 0;
      static IPoliticaNomina * crearPolitica(int, Nomina *);
};
class Nomina {
      float IRPF;
      float salarioBase;
      int opcion;
public:
      void setOpcion(int op) {opcion = op;}
      void setIRPF(float Im) {IRPF = Im;}
      void setSalarioBase(float salario) {salarioBase = salario;}
      float getSueldoNeto() {
          IPoliticaNomina *pPolitica = IPoliticaNomina::crearPolitica(opcion, this);
          return ( pPolitica -> getSueldoNeto() );
      float getSalarioBase() {return salarioBase;}
      float getIRPF() {return IRPF;}
};
class PoliticaBase : public IPoliticaNomina {
      friend IPoliticaNomina;
      Nomina *pNomina;
      PoliticaBase(Nomina * pNom): pNomina(pNom) {}
public:
      float getSueldoNeto(){
             return pNomina->getSalarioBase()*(1-(pNomina->getIRPF()/100));
      }
};
#define BONUS1.35
class PoliticaBonus : public IPoliticaNomina {
      friend IPoliticaNomina;
      Nomina *pNomina;
      PoliticaBonus(Nomina * pNom): pNomina(pNom) {}
public:
      float getSueldoNeto(){
             return pNomina->getSalarioBase()*BONUS*(1-(pNomina->getIRPF()/100));
};
```

ngeniería yd seño ndustrial
UNIVERSIDAD POLITÉCNICA DE MADRID ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDSUTRIAL
Departamento Electrónica, Automática e
Informática Industrial

APELLIDOS														
NOMBRE								Ν	0 N	Иа	t.			

ASIGNATURA: SISTEMAS INFORMÁTICOS INDUSTRIALES CURSO 4º **GRUPO** Diciembre 2013 Calificación

2. Problema de Sistemas Operativos (10 puntos - 25 minutos)

Se desea ejecutar sobre un sistema UNIX el fragmento de código adjunto.

En el código del programa se proyecta el fichero '/home/fich' en memoria como una zona de datos compartida.

Se pide:

1. Escribir el código para realizar la sincronización necesaria utilizando el mecanismo de semáforos. El código de sincronización debe hacer que ejecute primero el proceso padre, después el proceso hijo y así sucesivamente. (6 puntos)

El código de sincronización está indicado en el fragmento mediante el comentario: /* Código de sincronización */

- 2. Indicar qué valores escribe el proceso hijo por pantalla durante las cinco primeras iteraciones. Justificar razonadamente la respuesta. (2 puntos)
- 3. Explicar qué otros mecanismos de sincronización podrías utilizar para este caso. Justificar razonadamente la respuesta. (2 puntos)

Solución:

- 1. Indicado en rojo en el código. Como se aprecia en la solución, para que exista alternancia entre padre e hijo es necesario implementarlo con dos semáforos.
- 2. Los valores escritos en pantalla por el hijo son:

```
x: 0, y: 0
x: 1, y: 0
x: 3, y: 5
x: 1, y: 7
```

```
#define MAX IT 10
struct punto {
 int x;
 int y;
struct recta {
 struct punto pt1;
 struct punto pt2;
struct punto *pp;
struct recta *pr;
int main (void)
  int fd:
  struct stat bstat;
  int i;
  void *p;
  sem_t sem_padre, sem_hijo;
  sem init(&sem padre, 1, 1); // Padre 1° en ejecutar
  sem_init(&sem_hijo, 1, 0);
  fd = open ("/home/fich", O RDWR);
  fstat(fd, &bstat);
  p= mmap((c_addr_t) 0,bstat.st_size,PROT_READ |
             PROT WRITE, MAP SHARED, fd, 0);
  close (fd);
  if (fork() =! 0){
    pr = p;
    for ( i=0; i<=MAX IT; i++){</pre>
     sem wait(sem padre);
     pr-pt1.x = 3*i;
     pr->pt1.y =5*i;
     pr->pt2.x = 1;
     pr-pt2.y = 7*i;
     pr++; // Avanza hasta la siguiente estructura
     sem post(sem hijo);
  } else{
    pp = p;
    for ( i=0; i<=MAX IT; i++){</pre>
      sem wait(sem hijo);
      printf("x:%d, y:%d\n",pp->x, pp->y);
      pp++; // Avanza hasta la siguiente estructura
      sem post(sem padre);
    1
  }
```

3. No todos los mecanismos vistos para resolver el problema de la sección crítica sirven en este caso ya que se requiere alternancia entre procesos. Entre las soluciones alternativas estaría utilizar mutex y variables condicionales (pero transformando el código a procesos ligeros), paso de mensajes mediante una cola con capacidad de un único mensaje que haría de testigo entre los dos procesos, o tuberías siguiendo el mismo funcionamiento.

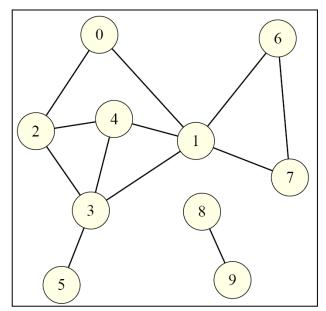
escuela técnica superior de ngeniería seño ndustrial universidad politécnica de madrid
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDSUTRIAL
Departamento Electrónica, Automática e Informática Industrial

APELLIDOS			
NOMBRE			Nº Mat.
			Calificación
ASIGNATURA:	SISTEMAS INFORMÁ	TICOS INDUSTRIALE	s
CURSO 4º	GRUPO	Diciembre 2	2013

3. Problema de algoritmia (10 puntos - 20 minutos)

El grafo de la figura representa una configuración de trayectorias en una red local de transporte de una planta. Los números representan índices. Se pide:

- 1. Número de clique $(\omega(G))$ y grado del grafo $(\Delta(G))$, justificando la respuesta
- 2. Resultado de lanzar un algoritmo *primero-en-profundidad* (sin indicar un vértice de partida). Indique exclusivamente el sello de tiempo inicial y final para cada nodo. Asuma que, a igualdad de criterio, el algoritmo siempre elige aquél vértice de numeración más baja.
- 3. Para implementar dicho algoritmo se emplea un contenedor *deque* (STL) que gestiona la frontera de la búsqueda. Implemente en C++ una función que permita mostrar en pantalla su contenido: a) mediante la función *copy*, b) recorriendo directamente el contenedor con iteradores.

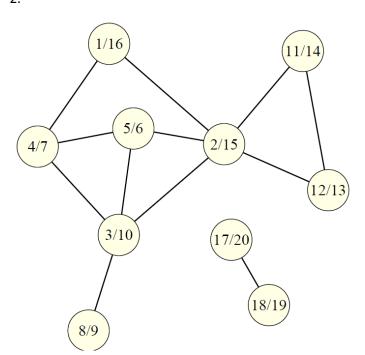


4. Implemente en C++ un algoritmo recursivo para computar el término n-ésimo de la sucesión $2a_n = 3a_{n-1} + 2a_{n-2}$ con $a_0 = 1$, $a_1 = 2$. Haga una estimación del número de llamadas a la función recursiva en función de n e indique, si fuera posible, alguna forma de optimizarlo.

Solucion:

1. $\omega(G)=3$ (grafo triangular) , $\Delta(G)=5$ (deg(1)=5)

2.



```
3. a)
      deque<int> mydeque;
      copy(mydeque.begin(), mydeque.end(), ostream_iterator<int>(cout, ""));
 b)
      deque<int>::iterator it;
      for(it=mydeque.begin(); it!=mydeque.end(); it++){
            cout<<*it<<" ";</pre>
      }
4.
long long int sucesion(int n){
      if(n==0) return 1;
      if(n==1) return 2;
      long long int r1=sucesion(n-1);
      long long int r2=sucesion(n-2);
      return (3/2)*r1+r2;
}
Árbol binario con n-2 niveles, 2<sup>n-2</sup> nodos aprox., de complejidad exponencial en
tiempo. Para obtener complejidad polinomial hay que memorizar el resultado de
sucesion(n-k) en cada nivel k.
long long int sucesion m(int n){
      if(n==0) return 1;
      if(n==1) return 2;
      long long int r1, r2;
      if(memoi[n-1]>0)
                  r1=memoi[n-1];
      else{
            r1=sucesion_m(n-1);
            memoi[n-1]=r1;
      if(memoi[n-2]>0)
                  r2=memoi[n-2];
      else{
            r2=sucesion_m(n-2);
            memoi[n-2]=r2;
      return (3/2)*r1+r2;
}
int main(){
      for(int n=0; n<N; n++){</pre>
            memoi.clear();
            for(int j=0; j<N; j++){</pre>
                  memoi.push_back(-1);
            }
            cout<<"s("<<n<<")"<<":"<<sucesion_m(n)<<endl;</pre>
      }
}
```