ICOM4015 Spring 2001 Exámen III 5/10/01

Nombre: _____ **Sección:** _____

¡Anota tu nombre y número de sección en <u>todas las hojas</u> del examen AHORA! (penalidad de 5 puntos)

Tienes 2 horas para completar tres problemas. Lee cuidadosamente todo el examen antes de empezar a trabajar. Muestra todo el trabajo conducente a tu contestación. Podrás recibir crédito parcial por contestaciones parciales siempre y cuando muestres tu trabajo por escrito. Usa tu tiempo inteligentemente. Exito!

ICOM 4015 Staff

1	33
2	33
3	34
Total	100

Nombre: _____ **Sección:** _____

Problema 1. (33 puntos) Manejo de listas.

Dada la siguiente estructura y definición codifique las funciones utilizando las declaraciones provistas:

```
struct node {
  int info;
  struct node *next;
};

typedef struct node *nodeptr;
```

(a) **(5 puntos)** Escriba una función que borre el último nodo de una lista; si la lista esta vacía devuelve **true** de lo contrario devuelve **false**.

```
bool dellast(nodeptr p) {
  nodeptr q;
  if (p == NULL) return true;

  q = p -> next;
  p -> next = q -> next;
  delete p;
  p = NULL;

  return false;
} /* end dellast */
```

(b) **(11 puntos)** Utilizar la función anterior para borrar una lista completa.

```
void clean(nodeptr p) {
  nodeptr q = p; nodeptr z;

  while (p -> next != NULL) {
    while (q -> next != NULL) {
      z = q;
      q = q -> next;
    }
    q = p;
    dellast(z);
  }
  p = NULL;
  z = NULL;
  delete q;
  q = NULL;
} /* end clean */
```

¡OJO! Problema 1 continúa en la próxima página ...

Nombre: _____ **Sección:** _____

(c) **(11 puntos)** Buscar un nodo en una lista sorteada ascendentemente. La función devuelve **true** si lo encuentra y **false** lo contrario. Si lo encuentra devuelve el pointer donde esta el nodo vía parámetro. Si no lo encuentra devuelve el pointer del último nodo menor vía parámetro.

```
bool search(nodeptr p, nodeptr q, int px) {

  nodeptr before = NULL;
  bool menor = true;

  q = p;
  while ((q != NULL) && (menor) ) {
    if (q -> info >= px) menor = false;
    else {
      before = q;
      q = q -> next;
    }
  }

  if (q == NULL) {
    q = before;
    return false;
  } else
      if (q -> info == px) return true;
      else {
        q = before;
        return false;
      }
} /* end search */
```

¡OJO! Problema 1 continúa en la próxima página ...

Nombre: _____ **Sección:** _____

(d) **(6 puntos)** Utilizando el resultado de la función anterior, inserte un nuevo nodo en la lista sorteada ascendentemente en el lugar que le corresponde. Recuerde que la estructura es la dada al principio del problema. La lista puede estar vacía, y si es así debe devolver el pointer del primer nodo.

```
nodeptr insert(nodeptr pos, int x) {

  nodeptr q;

  q = new node;
  q -> next = NULL;
  q -> info = x;


  if (pos == NULL) return q;

  q -> next = pos -> next;
  pos -> next = q;
  return NULL;

} /* end insert */
```

Nombre: _____ **Sección:** _____

Problema 2. Clases (33 puntos)

(a) **(10 puntos)** Una estructura permite al programador crear un conjunto con elementos de diferentes tipos de datos. De la misma forma, una clase puede cumplir con el mismo objetivo. A continuación se presenta una definición de una estructura:

```
struct point {
     double x, y;
};

void print(point *c)
{ cout << "(" << c.x << "," << c.y << ")"; }

void init(point *c, double u, double v) { c.x = u; c.y = v; }

void plus(point *c) {
     c.x += c.y;
     c.y += c.x;
}
```

Rescriba la estructura anterior como una clase. La clase resultante debe tener el mismo nombre de la estructura y estar compuesta por los mismos elementos.

```
class point {
  public:
     void print() { cout << "(" << x << "," << y << ")"; }
     void init(double u, double v) { x = u; y = v; }
     void plus(point c);

  private:
     double x, y;
};

void point::plus(point c) {
     x += c.y;
     y += c.x;
}
```

¡OJO! Problema 2 continúa en la próxima página ...

Nombre: _____ **Sección:** _____

(b) **(8 puntos)** Dado que los objetos de una clase también se pueden acceder utilizando los *Member Access Operators* (`dot operator(.)` y `arrow operator(->).` Complete el siguiente recuadro, colocando en las casillas vacías la expresión correcta.

La definición de la clase ***point*** es la utilizada en el punto a del examen.

Declaraciones y asignaciones

```
int main()
{
  point w, *p = &w;
  point v[15];
  w.x = 1;
  w.y = 4;
  v[0] = w;

return 0;
}
```

Expresión	Expresión Equivalente	Valor asignado
`w.x`	`P -> x`	`1`
`w.y`	`P -> y`	`4`
`V[0].x`	`P -> x`	`1`
`(*p).x`	`P -> y`	`4`

(c) **(15 puntos)** Cree una clase que permita representar un numero fraccionario como un objeto tipo *Fraction*, el cual estará compuesto por dos elementos enteros que se llamaran numerador y denominador. Además implemente los siguientes métodos:

- *printFraction*: Imprime la fracción.
- ***setFraction***: Consiste en darle valores al numerador y al denominador de una fracción.
- ***incrementFraction***: Consiste en sumar un uno a la fracción.
- ***addTo***: Consiste en sumar dos fracciones cualquiera.

Nota: En la pagina siguiente tiene todo el espacio que necesita para codificar su respuesta.

¡OJO! Problema 2 continúa en la próxima página ...

ICOM4015 Spring 2001 Exámen III 5/10/01

Nombre: _____ **Sección:** _____

```
class Fraction {
    public:
      fraction();
      void printNumber();
      void setFraction(int numer, int denom);
      void incrementFraction ();
      void addTo();

    private:
      int numerator;
      int denominator;
 };

Fraction::Fraction()
{
  numerator = 0;
  denominator = 1;
}

void Fraction::printFraction()
{
  cout << numerator << "/" << denominator;
}

void Fraction::setFraction(int numer, int denom)
{
  numerator = numer;
  denominator = denom;
  return;
}

void Fraction::incrementFraction()
{
  numerator += denominator;
  return;
}

void Fraction::addTo(const Fraction& fracc)
{
  if (denominator == fracc.denominator)
     numerator += fracc.numerator;
  else {
     numerator = (numerator * fracc.denominator) +
                 (fracc.numerator * denominator)
     denominator *= fracc.denominator;
  }
  return;
}
```