Segundo Parcial de Programación 1

Instituto de Computación

Noviembre 2009

Observaciones:

- Todos los programas o fragmentos de programas deben ser escritos en el lenguaje Pascal tal como fue dado en el curso. A grandes rasgos, este es el Pascal estándar con algunos agregados, a saber:
  - Utilización de else en la instrucción case. Si no se especifica else se ejecutará la sentencia siguiente al case en caso de no coincidir con ninguna de las etiquetas.
  - Evaluación por circuito corto de los operadores booleanos AND y OR.
- En todos los problemas se evaluará, además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso. De esta manera se restarán puntos entre otros conceptos por: mala o nula indentación, mala utilización de las estructuras de control, código confuso e innecesariamente largo, programas ineficientes, utilización de variables globales, pasaje incorrecto de parámetros, etc. No obstante, por razones prácticas no exigimos que incluya comentarios en los códigos que escriba en la prueba.
- Escriba su nombre completo y cédula en todas las hojas.
- Numere todas las hojas entregadas y escriba en todas la cantidad total de hojas.
- Escriba de un solo lado de cada hoja, un ejercicio por hoja.

Ejercicio 1

Dadas las declaraciones siguientes, donde cada una de las constantes tiene un valor apropiado:

```
const MaxEnteros = ... ;
const MaxIndices = ... ;
const MaxTabla = ... ;

type arre_enteros = array [1..MaxEnteros] of Integer;
type arre_indices = record
                        indices : array [1..MaxIndices] of 1..MaxEnteros;
                        tope : 0 .. MaxIndices;
                    end;
type tabla_enteros = array [1..MaxTabla] of arre_enteros;
```

a) Escribir el procedimiento *indices_de_primos* que, dado un arreglo de enteros mayores o iguales a 0 llamado *entrada*, guarde en un arreglo con tope llamado *salida*, los índices *i* de aquellos enteros de *entrada* que cumplan todas y cada una de las tres condiciones siguientes:

- entrada[i] es primo
- entrada[i+1] es par
- entrada[i-1] es par

```
procedure indices_de_primos(entrada: arre_enteros; VAR salida: arre_indices);
```

Ejemplo: si entrada contiene 2 5 3 1 3 4 7 8 2 2 5 0
los índices a guardar son: 7 9 11

Se puede usar la función *esPrimo* que devuelve *true* si *numero* es primo y *false* en caso contrario. No es necesario implementarla.

```
function esPrimo(numero: integer): boolean;
```

b) Escribir la función *al_menos_p* que, dada una tabla de tipo *tabla_enteros* y un entero *p*, devuelve *true* si para cada elemento de la tabla de tipo *arre_enteros* se cumple que existen al menos *p* números mayores que 100, *false* en caso contrario.

```
function al_menos_p (tabla : tabla_enteros; p : Integer) : Boolean;
```

Ejercicio 2

a) Con el procedimiento *agregar_al_final* se pretende solucionar el problema de insertar un elemento al final en una lista. Determine si la solución es correcta y, en caso de que no lo sea, justifique brevemente su respuesta.

```
type
    listaChar = ^celda;

    celda = record
              elemento: Char;
              siguiente: listaChar;
            end;

   procedure agregar_al_final(var l: listaChar; elem: Char);
   var p,q : listaChar;
   begin
      new(p);
      p^.elemento:=  elem;
      p^.siguiente:= nil;
      q:= l;
      while q^.siguiente <> nil do
         q:= q^.siguiente;
         q^.siguiente:= p
      end;
```

b) Escriba una función en Pascal que, dada una lista de caracteres, devuelva su largo.

```
function largo(l: listaChar): Integer;
```

Ejercicio 3

La Corte Electoral decidió desarrollar un sistema que gestione las votaciones en las mesas receptoras de votos para ser usado en las elecciones nacionales. Para ello, definió las estructuras que se presentan a continuación, donde una urna contiene los votos realizados. El voto puede ser en blanco, válido o anulado.

```
CONST
   MAX_VOTOS = ...;
   MAX_LISTAS = ...;
TYPE
   tipoVoto = (blanco, valido, anulado);
   voto = record
             case tipo : tipoVoto of
                blanco  : ();
                valido  : (
                             lista : integer;
                             siPlebiscito : boolean;
                          );
                anulado : ( causa : char);
          end;
   tipoUrna = record
                votos : array [1..MAX_VOTOS] of voto;
                cantVotos : 0..MAX_VOTOS;
              end;
```

a) Implemente el procedimiento *realizarVotoValido* que, dada una *lista* y un parámetro *siPlebiscito* para indicar si hay papeleta o no, agrega un voto válido en la urna. Asuma que siempre hay lugar en la urna para agregar un voto más.

```
procedure realizarVotoValido(lista: Integer; siPlebiscito: Boolean;
                             VAR urna: tipoUrna);
```

b) Para realizar el escrutinio de los votos, se definen las siguientes estructuras de datos:

```
       votacionLista = record
                          lista : Integer;
                          cantVotos : Integer;
                       end;
       tipoEscrutinio = record
                           cantBlanco : Integer;
                           cantAnulado : Integer;
                           cantSiPlebiscito : Integer;
                           votacionListas : array [1..MAX_LISTAS] of votacionLista;
                        end;
```

Implemente el procedimiento *realizarEscrutinio* que, dada una *urna*, devuelve en *escrutinio* el conteo total de votos, es decir, cuenta la cantidad de votos en blanco, la cantidad de votos anulados, la cantidad de votos por el plebiscito y las cantidades de votos para cada lista. Debe suponer que *escrutinio* se recibe inicializado de forma que los campos *cantBlanco*, *cantAnulado* y *cantSiPlebiscito* valen cero y el campo *votacionListas* contiene todas las listas que se pueden votar con *cantVotos* igual a cero.

```
procedure realizarEscrutinio(urna: tipoUrna; VAR escrutinio: tipoEscrutinio);
```