CERTAMEN #1 INF-154
Pauta de Corrección

Wenceslao Palma <wenceslao.palma@ucv.cl>

1. Realice el ruteo de los sgtes programas:

(a) (5 ptos.)

```
int n=5;
main(){
int a=3,b=4,*p;
     p = &b;
    *p = f(&a);
    printf("%d %d %d\n",*p,b,n++);
}
int f(int *p){
    *p = *p + ++n%2;
    return (*p);
}
```

(b) (5 ptos.)

```
int n=5;
main(){
    int x=4,y=0,z;
    z=f(&x,&y);
    printf("%d %d %d\n",x,y,z);
}
int f(int *p, int *q){
     *p = *p + 1;
     p = q;
     *p = ++n;
     return *p;
}
```

R.:

(a) (5 ptos.)

main			f			global	
a	3-3	200	p	200	500	n	5-6	100
b	4-3	300						
p	300	400						

salida (printf): 3,3,6

(b) (5 ptos.)

main			f			global	
x	4-5	200	p	200-300	500	n	5	100
y	0-6	300	q	300	600			
z	6	400						

salida (printf): 5,6,6

2. (30 ptos.) Un número de nueve dígitos se dice deleitable si:

   (a) contiene exactamente los dígitos entre 1 y 9, una vez cada uno, y

   (b) los números creados tomando los n primeros dígitos ($1 \leq n \leq 9$) son cada uno de ellos divisibles por n, de tal forma que el primer dígito será divisible por 1 (siempre lo será), los dos primeros

dígitos forman un número divisible por 2, los tres primeros forman un número divisible por 3 y así sucesivamente. Por ejemplo: si consideramos el número 123456789.

```
1:1   = 1
12:2  = 6
123:3 = 41
```

sin embargo 1234 no es divisible por 4.

Escriba la función **int deleitable(long int)** que permite determinar si un número es o no deleitable. R.:

```
Puntaje:
        uso de funciones:  5 ptos.
        verificar cantidad de dígitos: 8 ptos.
        verificar contiene digitos tan solo una vez:  8 ptos.
        verificar divisibilidad: 9 ptos.

 int deleitable(long int n){
     if ((cantidadDigitos(n)==9) && noContieneCero(n))
        if contieneDigitosUnaVez(n)
            if divisible(n)
                return 1;
            else
                return 2;
        else
            return 0;
     else
         return 0;
 }

 int cantidadDigitos(long int n){
    int l;
    int digito;

    l=0;
    while (n>0){
        n=n/10;
        l++;
    }

    if (l==9)
      return 9;
    else
      return 0;
 }

 int noContieneCero(long int n){
     int digito;

     while ((n>0) && (digito!=0)){
        digito = n%10;
        n=n/10;
    }
    if (digito==0)
       return 0;
```

```
   else
      return 1;
}

int contieneDigitosUnaVez(long int n){
  int cont[9]={0,0,0,0,0,0,0,0,0};

  while (n>0){
      digito = n%10;
      cont[digito]++;
      n=n/10;
  }

  estaUnaVez=1;
  while ((i<=8) && (estaUnaVez)){
     if (cont[i]==1)
         i++;
     else
         estaUnaVez = 0;
  }

  if (estaUnaVez)
    return 1;
  else
    return 0;
}

int divisible(long int n){
    int esDivisible,i=9;

    esDivisible = 1;
    while ((n>0) && (esDivisible)){
       if (n%i==0){
          n=n/10;
          i--;
       }else
           esDivisible=0;
    }
    if esDivisible
       return 1;
    else
       return 0;
}
```

3. (20 ptos.) Escriba un programa que dado un texto almacenado en un arreglo bidimensional calcule:

   - la cantidad de palabras diferentes dentro del texto.
   - la frecuencia de cada palabra dentro del texto.

   Considere que el arreglo bidimensional donde se almacena el texto es definido de la sgte manera: **char texto[10][100];** y las palabras se encuentran separadas por un espacio.

```
 Puntaje:
         cantidad palabras diferentes: 10 ptos.
         frecuencia de cada palabra: 10 ptos.

#include <stdio.h>
#include <string.h>

void frecuencia(char texto[][50],int n, int cantidadPalabras);
main(){
 char texto[2][50]={"hola que tal festival","un dos tres que festival"};
 char textoAux1[2][50]={"hola que tal festival","un dos tres que festival"};
 char textoAux2[2][50]={"hola que tal festival","un dos tres que festival"};

 int i,cantidadPalabras;

 cantidadPalabras=cantPalabras(texto,2);
 printf("Cantidad de palabras: [%d]\n",cantidadPalabras);
 printf("Cantidad de palabras diferentes: [%d]\n",
         cantidadPalabrasDiferentes(textoAux1,2,cantidadPalabras));
 frecuencia(textoAux2,2,cantidadPalabras);

}


int cantPalabras(char texto[][50],int n){
 int i,cont=0;;
 char token[] = " ";
 char *palabra=NULL;

 for(i=0;i<n;i++){
     palabra = strtok(texto[i],token);
     while (palabra != NULL) {
            palabra = strtok(NULL,token);
            cont++;
     }
 }
 return cont;
}


int cantidadPalabrasDiferentes(char texto[][50],int n, int cantidadPalabras){
 char *diccionario[cantidadPalabras];
 int i,j,k=0,esta;
 int cantPalabrasDifs=0;
 char token[] = " ";
 char *palabra=NULL;

 for(i=0;i<n;i++){
     palabra = strtok(texto[i],token);
     while (palabra != NULL){
           diccionario[k++] = palabra;
           palabra = strtok(NULL,token);
     }
 }

 for(i=0;i<cantidadPalabras;i++){
```

```
   esta=0;
   for(j=i+1;(j<cantidadPalabras) && (!esta);j++){
        if (strcmp(diccionario[i],diccionario[j])==0)
                esta=1;
   }
   if (!esta)
     cantPalabrasDifs++;
 }
 return cantPalabrasDifs;
}

void frecuencia(char texto[][50],int n, int cantidadPalabras){
 char *diccionario[cantidadPalabras];
 int i,j,k=0,esta,freq;
 int cantPalabrasDifs=0;
 char token[] = " ";
 char *palabra=NULL;

 for(i=0;i<n;i++){
     palabra = strtok(texto[i],token);
     while (palabra != NULL){
          diccionario[k++] = palabra;
          palabra = strtok(NULL,token);
     }
 }

 printf("Frecuencia:\n");
 for(i=0;i<cantidadPalabras;i++){
   esta=0;
   freq=1;
   if (strcmp(diccionario[i]," ")!=0){
       for(j=i+1;j<cantidadPalabras;j++){
           if (strcmp(diccionario[i],diccionario[j])==0){
                freq++;
                esta=1;
                strcpy(diccionario[j]," ");
           }
       }
      printf("[%s]--->[%d]\n",diccionario[i],freq);
   }
 }
}
```