Universidad Nacional de Ingeniería
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación
Introducción a la Programación Universidad

Tercera Práctica Calificada

CC102-AB **Ciclo: 2017-1**

Normas:

1. El alumno entregará esta hoja de examen debidamente llenada con sus datos.
2. La solución de la prueba se guardarán en **Escritorio**, carpeta: **ApellidoNombreCodigo** (sin espacios en blanco), la pregunta **n** se guardará en el archivo: **n.c** (n = 1, 2, ..).
3. No se permite: El uso de celulares, internet, USB, ingresar después de 15 min. de iniciado el examen ni salir antes de la hora de finalización.
4. Todo acto anti-ético será amonestado y registrado en el historial del estudiante.

Apellidos : _____ Nombres : _____
Sección : ____ Grupo: ___

1. [5 ptos.] Almacene los valores: 2, 3, . . . , 2017 en un arreglo de enteros del tipo short. Luego, utilizando la criba de Eratóstenes, imprima los primos encontrados en el arreglo y la dirección de memoria en la que se encuentra. Por ejemplo: 2 está en la dirección 0x7ffed9b65530 3 está en la dirección 0x7ffed9b65532 5 está en la dirección 0x7ffed9b65536 . . . 2017 está en la dirección 0x7ffed9b664ee Sugerencia: utilizar el siguiente pseudocódigo.

2. [5 ptos.] Se tiene una matriz de 1 y 0s de F = 10 filas por C = 10 columnas llamada laberinto, ejemplo:

```
int laberinto[F][C] = { {1,1,1,1,1,1,1,1,1,1},
                        {0,0,0,0,1,0,0,0,0,1},
                        {1,1,1,0,1,0,1,1,0,1},
                        {1,1,1,0,1,0,1,1,0,1},
                        {1,0,0,0,1,0,0,1,0,1},
                        {1,0,1,1,1,1,0,1,0,1},
                        {1,0,1,1,0,0,0,1,0,1},
                        {1,0,0,0,0,1,1,1,0,1},
                        {1,1,1,1,1,1,1,1,0,0},
                        {1,1,1,1,1,1,1,1,1,1} };
```

Los 1 representan al caracter ‘#’ y los 0s representan al espacio en blanco ‘ ‘. Escriba un programa que grafique laberinto[F][C]; el resultado será:

```
##########
      #     #
### # ## #
### # ## #
#    #   # #
# #### # #
# ##    # #
#     ### #
########
##########
```

Indicaciones:
Defina un arreglo de punteros, llamado *pFilas[F] de F elementos, que apuntarán a las direcciones de los inicios de cada fila de la matriz laberinto: pFilas[i] = laberinto[i];
Recorra cada fila *pFilas[i] con un puntero *q:

```
for(q=pFilas[i]; q<pFilas[i]+F; q++) { // grafique}
```

3. [5 ptos.] Escribir un programa que ingrese una cadena y un caracter cualquiera, luego cuente la cantidad de veces que se repite el carácter dentro de la la cadena.

4. [5 ptos.]Escribir un programa que lea el número total de compras de varias marcas de celulares, luego hacer un gráfico de barras (horizontal con '*').

Secciones CD

1. [5 ptos.] Escriba un programa que en la main():
Defina, por ejemplo:
int m = 4, n = 3; mm[4][3] = { 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4}, *pmm = &mm[0][0];
Llame a la función: reporte(pmm, m, n); // reporta mm
Llame a la función: suma2(pmm, m, n); // suma 2 a cada elemento de mm.
Llame a la función: reporte(pmm, m, n);

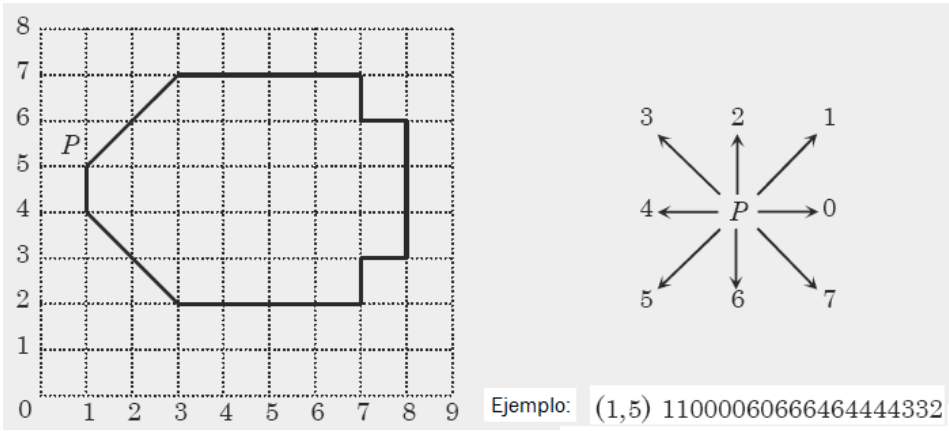
2. [5 ptos.] Escriba un programa que en la main():
Defina un arreglo arr[N] de N elementos y asigne valores.
Llame a la función: reporte(arr, N);
Llame a la función: ordenar(arr, N);
La cual ordena ASCENDENTEMENTE a arr haciendo comparaciones desde el inicio de arr y desde el final de la misma. Para ello utiliza dos punteros, uno que apunte al inicio del vector llamado *p y luego asciende, y otro que apunta al final del vector llamado *q y luego desciende. Complete la función:

```
void ordenar(int *arr, int n)        {
    int *p, *q;                                  // *p y *q
    int aux;
    while( !(estaOrdenado(arr, n))) {
            for(p=arr, q=arr+N-1; p<arr+N-1; p++, q--) {
                    if (*p > *(p+1))             // intercambiar datos
                    if (*(q-1) > *q)             // intercambiar datos
            }
    }
}

int estaOrdenado(int *arr, int pos) {            // verifica si arr está ordenado
    // si está ordenado retorna 1, si no 0.
}
```

Llame a la función: reporte(arr, N);

3. [5 ptos.] Las cadenas de *Freeman* representan objetos a partir de una codificación de sus entornos. El principio es que cualquier objeto puede ser encuadrado en un plano de mallas, y que a partir de cualquier punto de este plano nos podamos mover en ocho posiciones diferentes



Si se asume que los desplazamientos horizontales y verticales contribuyen en 1 unidad y los movimientos diagonales contribuyen en 1.42 unidades. Escribir un programa que lea una codificación de *Freeman*, e imprima el perímetro de la figura representada por la codificación; por ejemplo, la codificación “03245” tiene una perímetro de 5.84.

4. [5 ptos.] Para ingresar datos por teclado, se dan instrucciones al operador; pero si no es obediente, tendremos dificultades, por ejemplo, si se pide:

   Ingrese un entero mayor que 2: __ // El operador teclea: **tres**<enter> ingresó caracteres

   Para resolver este problema, se solicita repetidamente la lectura del dato hasta que cumpla la especificación (a eso se le llama **validar** el dato). Escriba un programa que lea un número (el operador puede ingresar caracteres) y valide que sea “entero mayor que 2”.