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.

```
// Programa AB1.c
#include <stdio.h>
#define max 2017
// Declarando prototipos de funciones
void   generar (short * p);
void   eratostenes (short * p);
void   mostrar (short * p);

int    main ( )
{
       short int criba[max];
       generar (criba);
       eratostenes (criba);
       mostrar (criba);
}

void   generar (short * p)
{
       int i = 2;
       while (i <= max) *p++ = i++;
       *p = -1;
}

void   eratostenes (short * p)
{
       int i = 1, j;
       while (++i*i <= max)
              if (*(p+i-2) > 0)
                     for (j = i*i ; j <= max ; j += i)
                            *(p+j-2) = 0;
}
```

```
void   mostrar (short * p)
{
        while (*p >= 0)
        {
                if (*p > 0) printf ("\n%5d está en la dirección %p",*p,p);
                p++;
        }
        printf ("\n\n");
}
```

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}
```

```
// Programa AB2.c
#include <stdio.h>
#define F 10
#define C 10

void main(void)
{
```

```
	int i;
	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} };
	int *pFilas[F];
	int *q;

	// ingresamos todos las direcciones de los inicios de las filas de la matriz
	for(i=0; i<F; i++)
		// pFilas[i] = (int *)laberinto + i*C;
	pFilas[i] = laberinto[i];

	// mostramos la matriz
	for(i=0; i<F; i++)
	{
		for(q=pFilas[i]; q<pFilas[i]+F; q++)
			printf("%c", (*q==1?'#':' ') );
		printf("\n");
	}
	printf("\n");
}
```

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.

```
// Programa AB3.c
#include <stdio.h>
#include <string.h>
int main (){
	char cadena [100];
	char car;
	int i,cuenta=0;
	printf ("Introduzca una cadena: ");
	fgets (cadena, 100, stdin);
	printf ("La cadena leida es: %s\n", cadena);
	printf ("Introduzca un caracter: ");
	car=getchar();
	printf ("El caracter ingresado es: %c\n", car);
	for(i=0;i<strlen(cadena);i++)
		if(cadena[i]==car)cuenta++;
	printf ("Son %d caracteres %c en la cadena\n", cuenta,car);
	return 0;
}
```

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 ‘*’).

```
//Programa AB4.c
```

```
#include<stdio.h>
#include <stdlib.h>
#define MAXLISTA 100
#define LONGE 100
int leerString(char cc[ ], int);

void main(void){
	char lista [MAXLISTA][LONGE];
	int n=0, i, j;
	puts("Ingreso de Número de Compras; para finalizar presione Enter");
	printf ("Numero de Compras: ");
	while (leerString(lista[n], LONGE) && n<MAXLISTA){
		printf ("Numero de Compras: ");
		n++;
	}
	for(i=0; i<n; i++) {
		putchar('\n');
		for(j=0; j< atoi(lista[i]); j++)
			putchar('*');
	}
	putchar('\n');
}
int leerString(char cc[ ], int n){
	int c, m=0;
	while((c=getchar())!=10) if(m<n-1)cc[m++]= c;
	cc[m] = '\0';
	return m;
}
```

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);

```
// CD1.c
#include<stdio.h>
void suma2(int *pmm, int m, int n);
void reporte(int *pmm, int m, int n);
void main(void){
	int m = 4, n = 3, mm[4][3] = { 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4}, *pmm = &mm[0][0];
	printf("Matriz inicial\n");
	reporte(pmm, m, n);
	suma2(pmm, m, n);
	printf("\nMatriz ordenada\n");
	reporte(pmm, m, n);
}
void reporte(int *pmm, int m, int n){
	int i, j;
	for(i=0; i<m; i++){
```

```
                for(j=0; j<n; j++) printf("%d  ", pmm[i*n+j]);
                printf("\n");
        }
}
void suma2(int *pmm, int m, int n){
        int *q;
        for(q = pmm; q < pmm+m*n; q++) *q += 2;
}
```

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.
}
```

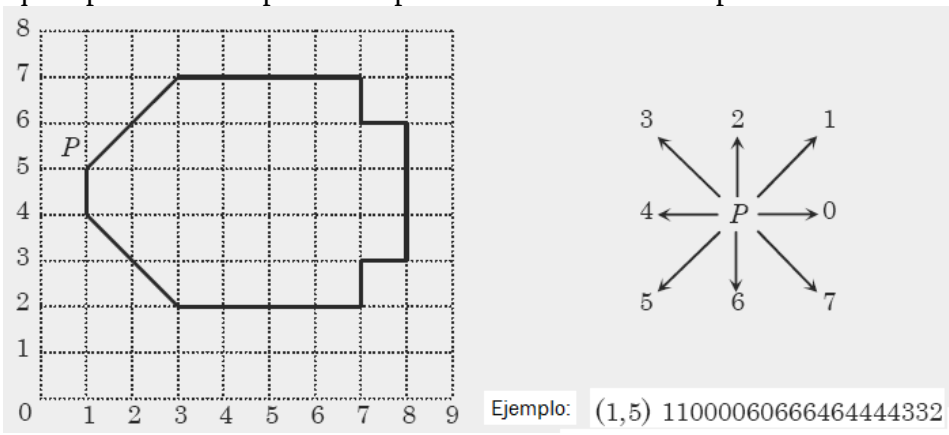
Llame a la función: reporte(arr, N);

```
// CD2.c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 10

int aleatorioEntre(int a, int b);
int estaOrdenado(int *arr, int pos);
void ordenar(int *arr, int elementos);
void mostrarArreglo(int *arr, int elementos);

void main(void)
{
        int datos[N];
        int i;
        srand(time(NULL));

        for(i=0; i<N; i++)
                datos[i]=aleatorioEntre(10,100);

        mostrarArreglo(datos,N);
```

```
	ordenar(datos,N);

	printf("\n\nArreglo ordenado:\n");
	mostrarArreglo(datos,N);
	printf("\n");
}

void mostrarArreglo(int *arr, int elementos)
{
	int i;
	for(i=0; i<elementos-1; i++)
		printf("%d, ",arr[i]);
	printf("%d",arr[i]);
}

int aleatorioEntre(int a, int b)
{
	return (a + rand() % (b-a+1) );
}

int estaOrdenado(int *arr, int pos)
{
	if(pos==1)                                    return 1;
	if(arr[pos-2]>arr[pos-1]) return 0;
	return estaOrdenado(arr,pos-1);
}

void ordenar(int *arr, int elementos)
{
	int *p, *q;
	int aux;
	while( !(estaOrdenado(arr,elementos)))
	{
		for(p=arr, q=arr+N-1; p<arr+N-1; p++, q--)
		{
			if (*p > *(p+1))
			{
				aux = *p;
				*p = *(p+1);
				*(p+1) =aux;
			}

			if (*(q-1) > *q)
			{
				aux = *q;
				*q = *(q-1);
				*(q-1) = aux;
			}
		}
	}
}
```

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.

```
// Programa CD3.c
#include<stdio.h>
#include <string.h>
#define MAXCOD 100
void main(void){
        char lista[MAXCOD];
        int i=0;
        float perimetro=0;
        printf("Ingrese codigo Freeman: "); scanf("%s",lista);
        for(i;i<strlen(lista);i++){
                if((lista[i]=='1')||(lista[i]=='3')||(lista[i]=='5')||(lista[i]=='7'))
                        perimetro=perimetro+ 1.42;
                else perimetro=perimetro+ 1.00;
}       }
        printf("\nPerimetro: %.2f\n", perimetro);
}
```

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”.

```
// CD4.c
#include <stdio.h>
#include <stdlib.h>
int leerString(char cc[ ], int n);
void main(void){
    char cc[11];
    int n;
    do {
            printf("Ingrese un entero mayor que 2: ");
            leerString(cc, 11);
            n = atoi(cc);
    } while(n<3);
```

```
    printf("Número ingresado: %d\n", n);
}
int leerString(char cc[ ], int n){
    int c, m=0;
    while((c=getchar())!=10) if(m<n-1)cc[m++]= c;
    cc[m] = '\0';
    return m;
}
```