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

Examen Final Ciclo: 2017-1 Duración: 2:00

Normas:

1. No se permite: El uso de celulares, internet, ni USB.
2. No se permite: Ingresar después de 15 min. de iniciado el examen; salir antes de la hora de finalización.
3. El alumno entregará esta hoja de examen debidamente llenada con sus datos
4. Todo acto anti-ético será amonestado y registrado en el historial del alumno.

---

1. [5 pts.] Implemente el siguiente prototipo de función:

void intercambio (int * x, int * y);

El cual intercambia los valores almacenados en las direcciones x e y. En la función principal, se debe definir un arreglo de 10 enteros y asignar a sus elementos valores aleatorios en el intervalo [0; 20]; finalmente se debe mostrar dicho arreglo y el arreglo que resulta de ordenar de menor a mayor el arreglo original, utilizando la función intercambio.

```
// 1.c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void intercambio (int * p1, int * p2);
int main () {
	short i = 0, j;
	int notas[10];
	srand(time(NULL));
	while (i < 10) {
	 notas[i] = rand()%21;
	 printf("%4d",notas[i++]);
	}
	printf("\n");
	for (j = 9 ; j > 0 ; j--)
	 for (i = 0 ; i < j ; i++)
		if (notas[i] > notas[i+1]) intercambio(notas+i+1, notas+i);
	i = 0;
	while (i < 10) printf("%4d",notas[i++]); printf("\n");
}
void intercambio (int * p1, int * p2){
	int aux = *p1;
	*p1 = *p2;
	*p2 = aux;
}
```

2. [5 pts.] Escriba un programa que lea una cadena de caracteres en inglés de tamaño no mayor a 30 formada sólo por letras o espacios y muestre dicha cadena escrita todo en mayúscula.

```
//2.c
#include <stdio.h>
#define max 30
void ingresar (char * p);
void capitalize (char * p);
void main ()
{
	char cad[max + 1];
```

```
	ingresar (cad);
	capitalize (cad);
}
void ingresar (char * p)
{
	int ch;
	printf("\n Ingrese una frase de a lo más %d caracteres (sólo letras o espacios): ", max);
	ch = getchar();
	while (ch != '\n')
	{
		*p++ = ch;
		ch = getchar();
	}
	*p = '\0';
}
void capitalize (char * p)
{
	printf (" ");
	while (*p != '\0')
	{
		if (65 <= *p && *p <= 90)
			printf ("%c",*p++);
		else if (97 <= *p && *p <= 122)
			printf ("%c",*p++-32);
		else {printf (" "); p++;}
	}
	printf ("\n\n");
}
```

3. [5 pts.] Escriba un programa que reserve memoria para números reales de un tamaño de 300. Ingrese números reales, finalice para 0. Redimensione la memoria con la cantidad de números ingresada. Imprima los números ingresados; finalmente mediante una función que calcule el promedio ajustado a 2 decimales de los números ingresados.

```
// 3.c
#include <stdio.h>
#include <stdlib.h>
void promedio(float *LISTADO,int cantidad) {
	int X=0 ;
	float suma=0;
	while(X<cantidad) {
		suma=suma+LISTADO[X];
		X=X+1;
	}
	printf("Promedio :\n");
	printf("%.2f ",(float)suma/cantidad);
	printf("\n");
}
void main(void){
	int i, contador = 0, lon=300;
	float n;
	float *p;
	p=(float *) malloc (lon*sizeof(float)) ;
	printf("Ingrese numeros reales (0 para terminar):\n");
	do {
		scanf("%f",&n);
		if(n) p[contador++] = n;
```

```
        } while(n);
        printf("Se han ingresado :%d\n",contador);
        p=(float *) realloc (p,contador*sizeof (float));
        for(i=0;i<contador;i++) printf("%.2f\n",p[i]);
        promedio(p,contador);
        free(p);
}
```

4. [5 pts.] Escriba un programa que:

Defina la estructura:

Estudiante {int codigo; char nombre[25]};

Defina dos arreglos tipo **Estudiante** con los datos:

est1[2] con los datos {{1, “Juan”}, {5, “José”}}

est2[3] con los datos {{3, “Rosita”}, {7, “María”}, {9, “Juana”}}

Como puede ver ambos arreglos están ordenados por código.

Imprima los datos de los dos arreglos ordenados por código; para los datos anteriores, la salida sería:

1 Juan
3 Rosita
5 José
7 María
9 Juana

Sugerencia (no es obligatorio juntar los dos arreglos y ordenarlos), puede hacer:

Iteración:

Inicie los índices de est1 y est2;

Repita: compare los códigos: Escriba la estructura de menor código y avánzela en 1;

Criterio de fin: Finaliza uno de los arreglos.

Proceso final: Escriba todos los datos remanentes del arreglo que no finalizó

```
// 4.c
#include<stdio.h>
typedef struct {
        int codigo;
        char nombre[25];
} Est;
void main(void){
    Est est1[2] = {{1, "Juan"},    {5, "José"}};
    Est est2[3] = {{3, "Rosita"}, {7, "María"}, {9, "Juana"}};
    int m = 2, n = 3, i=0, j=0;
    while(i<m && j<n) {
     if(est1[i].codigo <= est2[j].codigo){
        printf("%d  %s\n", est1[i].codigo, est1[i].nombre);
        i++;
     } else {
        printf("%d  %s\n", est2[j].codigo, est2[j].nombre);
        j++;
     }
    }
    if(i==m){
     while(j<n) {
        printf("%d  %s\n", est2[j].codigo, est2[j].nombre);
        j++;
     }
    } else {
     printf("%d  %s\n", est1[i].codigo, est1[i].nombre);
     i++;
    }
}
```