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

Examen Sustitutorio 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:

int verificar (int * dado, int n);

El cual retorna 1 si en un arreglo de 10 elementos apuntado por dado, contiene un cierto número n; caso contrario, retorna 0. En la función principal, se debe definir un arreglo de 10 enteros y asignar a sus elementos valores aleatorios en el intervalo [0; 20] y asignar aleatoriamente un valor para n en el mismo intervalo. Finalmente, verificar si n está en dicho arreglo o no, mostrando un mensaje en la pantalla.

```
// 1.c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int verificar (int * dado, int n);

int main ()
{
 short i = 0;
 int notas[10], N;
 srand(time(NULL));
 while (i < 10)
 {notas[i] = rand()%21; printf("%4d",notas[i++]);}
 N = rand()%21;

 if (verificar(notas,N)) printf ("\n %d se encuentra en el arreglo.\n",N);
 else printf ("\n %d NO se encuentra en el arreglo.\n",N);
}

int verificar (int * dado, int n)
{
 short i = 0;
 while (i < 10)
 if (dado[i++] == n) return 1;
 return 0;
}
```

2. [5 pts.] Escriba un programa que lea una palabra o frase en inglés de tamaño no mayor a 30 formada sólo por letras o espacios y muestre un mensaje diciendo si dicha palabra o frase es o no un palíndroma, e.g., las siguientes frases son palíndromas:

- rats live on no evil star
- step on no pets

```
// 2.c
#include <stdio.h>
#define max 30

void ingresar(char * p);
```

```
void compactificar(char * p, char * q, short * r);
int palindrome (char * q, short * r);

void main ()
{
 char cad[max + 1], k[max + 1];
 short N;

 ingresar(cad);
 compactificar(cad,k,&N);
 if (palindrome(k,&N) == 1) printf (" La frase ingresada es un palíndromo.\n\n");
 else printf (" La frase ingresada NO es un palíndromo.\n\n");

}

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 compactificar(char * p, char * q, short * r)
{
 *r = 0;
 while (*p != '\0')
 {
        if (65 <= *p && *p <= 90)
                {*q++ = *p++ + 32; *r += 1;}
        else if (97 <= *p && *p <= 122)
                {*q++ = *p++; *r += 1;}
        else p++;
 }
 *q = '\0';
}

int palindrome (char * q, short * r)
{
 short i;
 for (i = 0; i < *r/2; i++) if (q[i] == q[*r-i-1]) return 1;
 return 0;
}
```

3. [5 pts.] Escriba un programa que lea el número de estudiantes (0 para terminar) de cada salón correspondiente a un curso. Ud. no sabe cuántos salones necesita el curso, inicialmente asigne memoria dinámica para 8 salones. Al terminar de leer re-dimensionar el área a lo necesario y contar el número total de alumnos matriculados en el curso.

```
#include <stdio.h>
#include <stdlib.h>
void sumar(int *LISTADO,int cantidad) {
```

```
   int X=0 ;
   int suma=0;
   while(X<cantidad) {
      suma=suma+LISTADO[X];
      X=X+1;
   }
   printf("Total de alumnos del curso :\n");
   printf("%d",suma);
   printf("\n");
}
void main(void){
   int i, contador = 0, lon=8;
   int n;
   int *p;
   p=(int *) malloc (lon*sizeof(int)) ;
   printf("Ingrese numero de alumnos de un salon (0 para terminar):\n");
   do {
      scanf("%d",&n);
      if(n) p[contador++] = n;
   } while(n);
   printf("Se han ingresado %d salones\n",contador);
   p=(int *) realloc (p,contador*sizeof (int));
   for(i=0;i<contador;i++) printf("%d\n",p[i]);
   sumar(p,contador);
   free(p);
}
```

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

Defina la estructura:

Profe {int codigo; char nombre[25], int sueldo;};

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

profe[5] con los datos: {1, "Juan", 2000}, {3, "zoila", 1000}, {5, "Pedro", 1000}, {7, "Carlo", 1200}, {9, "Luis", 2000}}

nuevo[2] con los datos: {{3, "Zoila", 3000}, {7, "Carlos", 10000}}; el cual actualiza los datos de **profe** que tengan el mismo código. Note que **profe[]** y **nuevo[]** están ordenados por código.

Imprima los Datos Iniciales y Datos actualizados, para el ejemplo sería:

Datos Inciales			Datos Actualizados		
Código	Nombre	sueldo	Código	Nombre	sueldo
1	Juan	2000	1	Juan	2000
3	zoila	1000	3	Zoila	3000
5	Pedro	1000	5	Pedro	1000
7	Carlo	1200	7	Carlos	10000
9	Luis	2000	9	Luis	2000

```
// 4.c
#include<stdio.h>
#include<string.h>
typedef struct {
 int codigo;
 char nombre[25];
 int sueldo;
} Profe;
void main(void){
 Profe profe[5] = {{1, "Juan", 2000}, {3, "zoila", 1000}, {5, "Pedro", 1000}, {7, "Carlo", 1200}, {9, "Luis",
2000}};
 Profe nuevo[2] = {{3, "Zoila", 3000}, {7, "Carlos", 10000}};
```

```
 int m = 5, n = 2, i=0, j=0;
 printf("Datos Inciales\t\tDatos Actualizados\n");
 printf("Código\tNombre\tsueldo\tCódigo\tNombre\tsueldo\n");
 while(i<m && j<n){
        printf("%d\t%s\t%d\t",profe[i].codigo, profe[i].nombre, profe[i].sueldo);
        if (profe[i].codigo == nuevo[j].codigo){
                strcpy(profe[i].nombre, nuevo[j].nombre);
                profe[i].sueldo =  nuevo[j].sueldo;
                j++;
        }
        printf("%d\t%s\t%d\n",profe[i].codigo, profe[i].nombre, profe[i].sueldo);
        i++;
 }
 while(i<m){
        printf("%d\t%s\t%d\t",profe[i].codigo, profe[i].nombre, profe[i].sueldo);
        printf("%d\t%s\t%d\n",profe[i].codigo, profe[i].nombre, profe[i].sueldo);
        i++;
 }
}
```