UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERIA Y ARQUITECTURA
ESCUELA DE INGENIERIA DE SISTEMAS INFORMATICOS
ESTRUCTURA DE DATOS

SEGUNDO EVALUADO PRÁCTICO
CICLO II/2021

CARNE:_____NOMBRE:_____
FECHA: 11 de noviembre de 2021. GL: 03

Favor hacer un archivo comprimido con los archivos que se le piden en el evaluado y ponerle el nombre así:

EP2CARNETGL3

Donde: EP2 significa Evaluado Práctico 2
CARNET pondrán su respectivo carné.

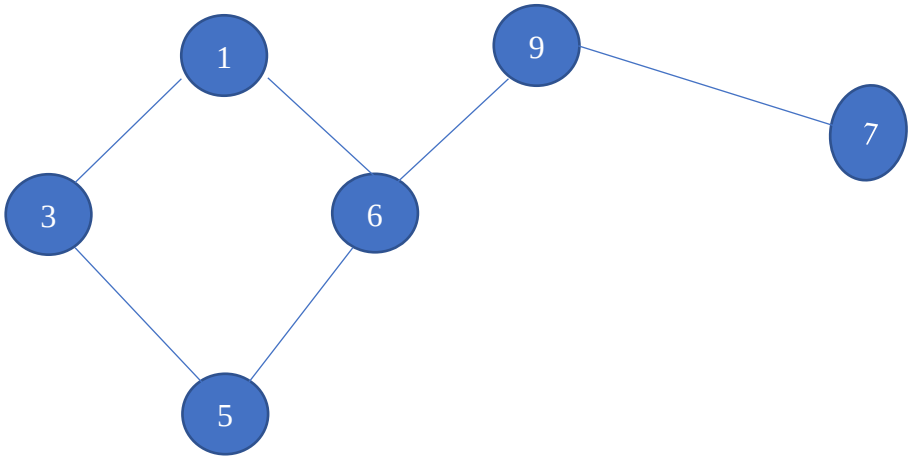
Tendrán 90 minutos para realizar el evaluado práctico y 5 minutos extra para que suban su archivo empaquetado al aula virtual.

Para la realización del evaluado práctico deberán programar una reunión en Google Meet con los docentes Inga. Patricia Haydée Estrada de López (patricia.estrada@ues.edu.sv) e Ing. Bladimir Díaz (bladimir.diaz@ues.edu.sv). Favor enviar la invitación el mismo día de la evaluación, a las 4:30 p.m.

En el momento de la evaluación deben de compartir cámara, mostrar su escritorio y grabar la reunión, para que al finalizar la evaluación compartan el video en Google Drive con los docentes antes mencionados. Favor utilizar el **correo institucional** (carnet@ues.edu.sv) para realizar lo que se pide.

¡ BUENA SUERTE !

1. (50%) Dado el grafo no dirigido siguiente, se pide hacer la matriz de adyacencia (25%) y el recorrido en anchura partiendo del vértice 5 (25%). Favor dejar constancia del proceso.




1. (50%) Escriba una función multiNodos que visite cada uno de los nodos de un árbol binario y multiplique por 5 todos los nodos que sean múltiplo de 7. Haga un programa en C que inserte 6 nodos en el árbol binario, lo recorra PreOrden, luego pruebe a la función anterior debiendo imprimir el resultado obtenido en PreOrden también.

| | |
|---|---|
| Función multiNodos | 3.5 |
| main | 1.5 |

Código:

```
#include <stdlib.h>
#include <stdio.h>
#define TRUE 1
#define FALSE 0
/* Estructuras y tipos */
typedef struct _nodo {
   int dato;
   struct _nodo *derecho;
   struct _nodo *izquierdo;
} tipoNodo;

typedef tipoNodo *pNodo;
typedef tipoNodo *Arbol;

void Insertar(Arbol *a, int dat);
int Vacio(Arbol r);
void InOrden(Arbol, void (*func)(int*));
void Podar(Arbol *a);
```

```
void Mostrar(int *d);
int main()
{
   Arbol ArbolInt=NULL;
   return 0;
}
/* Poda: borrar todos los nodos a partir de uno, incluido */
void Podar(Arbol *a)
{
   /* Algoritmo recursivo, recorrido en postorden */
   if(*a) {
      Podar(&(*a)->izquierdo); /* Podar izquierdo */
      Podar(&(*a)->derecho);   /* Podar derecho */
      free(*a);                /* Eliminar nodo */
      *a = NULL;
   }
}
/* Insertar un dato en el árbol ABB */
void Insertar(Arbol *a, int dat)
{
   pNodo padre = NULL;
   pNodo actual = *a;
/* Buscar el dato en el árbol, manteniendo un puntero al nodo padre */
   while(!Vacio(actual) && dat != actual->dato) {
      padre = actual;
      if(dat < actual->dato) actual = actual->izquierdo;
      else if(dat > actual->dato) actual = actual->derecho;
   }
/* Si se ha encontrado el elemento, regresar sin insertar */
   if(!Vacio(actual)) return;
   /* Si padre es NULL, entonces el árbol estaba vacío, el nuevo nodo será
      el nodo raiz */
   if(Vacio(padre)) {
      *a = (Arbol)malloc(sizeof(tipoNodo));
      (*a)->dato = dat;
      (*a)->izquierdo = (*a)->derecho = NULL;
   }
/* Si el dato es menor que el que contiene el nodo padre, lo insertamos en la rama
izquierda */
   else if(dat < padre->dato) {
      actual = (Arbol)malloc(sizeof(tipoNodo));
      padre->izquierdo = actual;
      actual->dato = dat;
      actual->izquierdo = actual->derecho = NULL;
   }
/* Si el dato es mayor que el que contiene el nodo padre, lo insertamos en la rama
derecha */
   else if(dat > padre->dato) {
      actual = (Arbol)malloc(sizeof(tipoNodo));
      padre->derecho = actual;
      actual->dato = dat;
```

```
      actual->izquierdo = actual->derecho = NULL;
  }
}
void InOrden(Arbol a, void (*func)(int*))
{
   if(a->izquierdo) InOrden(a->izquierdo, func);
   func(&(a->dato));
   if(a->derecho) InOrden(a->derecho, func);
}
int Vacio(Arbol r)
{
   return r==NULL;
}
void Mostrar(int *d)
{
   printf("%d, ", *d);
}
```