PROBLEMA 1

```
/*Estructuras de datos - LIS 2032-1
Nombre Completo:
ID:
Descripción breve del programa:*/
#include <stdio.h>
#include <stdlib.h>
struct Node{
	int data;
	struct Node* next;
};
struct Node* head;
void InsertFinal(){
	printf("Valor: ");int x; scanf("%d",&x);
	struct Node* temp=(struct Node*)malloc(sizeof(struct Node));
	struct Node* ptr; ptr=head;
	temp->data=x;
	temp->next=NULL;
	while(ptr->next!=NULL){
		ptr=ptr->next;
	}
	ptr->next=temp;
}
void InsertInicio(){
	printf("Valor: "); int x; scanf("%d",&x);
	struct Node* temp=(struct Node*)malloc(sizeof(struct Node));
	temp->data=x;
```

```
        temp->next=head;
        head = temp;
}
void InsertMiddle(){
        printf("  Posicion: "); int n; scanf("%d",&n);
        printf("  Valor: "); int x; scanf("%d",&x);
        struct Node* ptr= head;
        struct Node* temp1=(struct Node*)malloc(sizeof(struct Node));
        temp1->data=x;
        temp1->next=NULL;
        if(n==1){
                temp1->next=head;
                head = temp1;
                return;
        }
        struct Node* temp2 = head;
        int i;
        for(i=0; i<n-2; i++){
                temp2 = temp2 -> next;
        }
        temp1->next=temp2->next;
        temp2->next=temp1;


}
void Print(){
        struct Node* temp = head;
        printf("Lista: ");
```

```
	while(temp!= NULL){
		printf(" %d", temp->data);
		temp=temp->next;
	}
	printf("\n");
}
void DeleteMiddle(){
	printf("Posicion del dato a eliminar: ");
	int n; scanf("%d", n);
	struct Node* temp1=head;
	struct Node* temp2 = temp1->next; //nth Node
	if(head=NULL) {
		printf("Lista vacia\n");
		return;
	}
	if(n==1){
		head=temp1->next; //head now points to 2nd node
		free(temp1);
		return;
	}
	int i;
	for(i=0;i<n-2;i++){
		temp2=temp1;
		temp1=temp1->next;
		//temp1 points to (n-1)th node
	}
	temp2->next = temp1->next; //(n+1)th Node
	free(temp1); //delete temp2
```

```
            temp2=NULL;
}
void DeleteInicio(){
            if(head == NULL){
        printf("Lista vacia");
        return;
    }
    if(head->next==NULL){
            free(head);
            return;
            }
            struct Node* temp1=head;
            head=temp1->next;
            free(temp1);
}
void DeleteFinal(){
            if(head == NULL){
        printf("Lista vacia");
        return;
    }
    if(head->next==NULL){
            free(head);
            return;
            }
            struct Node* temp1=head; struct Node* temp2=head;
            while(temp1->next!=NULL){
                        temp2=temp1;
                        temp1=temp1->next;
            }
            temp2->next=NULL;
            free(temp1);
            temp1=NULL;
}
int main(){
            head=NULL;
            printf("\n LISTAS ENLAZADAS SIMPLES");
            printf("\n\t\t MENU");
            printf(" \n 1 : Insertar un nodo al inicio");
            printf(" \n 2 : Insertar un nodo al final");
            printf(" \n 3 : Insertar un nodo intermedio");
            printf(" \n 4 : Eliminar un nodo al inicio");
            printf(" \n 5 : Eliminar un nodo intermedio");
            printf(" \n 6 : Eliminar un nodo al final");
            printf(" \n 7 : Imprimir LISTA");
            printf(" \n 8 : SALIR");
int choice;
do{
            printf(" \n Introduzca la opcion: ");
            scanf("%d", &choice);
            switch(choice){
            case 1:{ // Case 1 Agrega un nodo al inicio
                        InsertInicio();
                        break;}
            case 2:{ // Case 2 Agrega un nodo al final
                        InsertFinal();
                        break;}
case 3:{ // Case 3 Agrega un nodo en posicion especifica
            InsertMiddle();
            break;}
case 4:{
            DeleteInicio(); // Case 4 Elimina un nodo al inicio
            break;
}
case 5:{
            DeleteMiddle(); // Case 5 Elimina un nodo intermedio
            break;
}
case 6:{
            DeleteFinal(); // Case 6 Elimina un nodo al final
            break;
}
case 7:{
            Print(); // Case 7 Imprime la lista
            break;
}
case 8:{ // Salir
            exit(0);
            break;
}
default:{
            printf(" This option doesn't exist'...\n");
            break;
}
}//end switch
```

```
 }while(choice!= 0);
} // end main
```

PROBLEMA 2

```
	printf("%d ",root->data);
	Inorden(root->right);
}
void Postorden(struct BstNode *root) {
	if(root == NULL) return;

	Postorden(root->left);
	Postorden(root->right);
	printf("%d ",root->data);
}

bool Search(struct BstNode* root, int data){
	if(root==NULL) return false;
	else if(root->data==data) return true;
	else if(data<=root->data) return Search(root->left, data);
	else return Search(root->right, data);
}
int FindMin(struct BstNode* root){
	if(root==NULL){
		printf("Arbol vacio\n");
		return -1;
	}
	while(root->left!=NULL){
		root=root->left;
```

```
        }
        return root->data;
}
int FindMax(struct BstNode* root){
        if(root==NULL){
                printf("Arbol vacio\n");
                return -1;
        }
        while(root->right!=NULL){
                root=root->right;
        }
        return root->data;
}
struct BstNode* Min(struct BstNode* root)
{
        while(root->left != NULL) root = root->left;
        return root;
}
struct BstNode* Delete(struct BstNode *root, int data) {
        if(root == NULL) return root;
        else if(data < root->data) root->left = Delete(root->left,data);
        else if (data > root->data) root->right = Delete(root->right,data);
        else {
                // no hijos
                if(root->left == NULL && root->right == NULL) {
                        free(root);
                        root = NULL;
                }
```

```
		//1 hijo
		else if(root->left == NULL) {
			struct BstNode *temp = root;
			root = root->right;
			free(temp);
		}
		else if(root->right == NULL) {
			struct BstNode *temp = root;
			root = root->left;
			free(temp);
		}
		// 2 hijos
		else {
			struct BstNode *temp = Min(root->right);
			root->data = temp->data;
			root->right = Delete(root->right,temp->data);
		}
	}
	return root;
}
int FindHeight(struct BstNode *root){
	if(root==NULL){
		return -1;
		return max(FindHeight(root->left), FindHeight(root->right))+1;
	}
}
void deleteTree(struct BstNode* root){
	if(root==NULL) return;
```

```
		deleteTree(root->left);
		deleteTree(root->right);
		printf("\nBorrando nodo: %d", root->data);
		free(root);
	}
	int main(){
		struct BstNode* root=NULL;
		root= Insert(root,15); //insertar
		root= Insert(root,10);
		root= Insert(root,20);
		root= Insert(root,25);
		root= Delete(root,25); //eliminar un nodo

		//buscar
		printf("Ingresa un numero a buscar: ");
		int num; scanf("%d",&num);
		if(Search(root, num)==true){
			printf("Numero encontrado!\n");
		}
		else{
			printf("Numero NO encontrado\n");
		}
		root = Delete(root,25);
		//valor min
		printf("valor minimo: ");
		FindMin(root);
		printf("\n");
		//valor max
```

```
        printf("valor maximo: ");
        FindMin(root);
        printf("\n");

        //imprimir preorden
        printf("Inorden: ");
        Preorden(root);
        printf("\n");
        //imprimir inorden
        printf("Inorden: ");
        Inorden(root);
        printf("\n");
        //imprimir postorden
        printf("Postorden: ");
        Postorden(root);
        printf("\n");

                //altura
        printf("Altura: ");
        FindHeight(root);
        printf("\n");

        //borrar arbol
        printf("Borrar arbol");
        deleteTree(root);
    root = NULL;
    printf("\n Arbol borrado");
```

```
}
```