```c
#include <stdio.h>
#include <stdlib.h>

// Alunos: Carlos Vinicius, João Victor

// Deverá ser implementado em C um sistema que armazene um número inteiro em uma árvore binária. E que também seja possível exibir o percurso escolhido pelo usuário, a saber: pré-ordem, em ordem pós-ordem. O sistema deverá possuir o seguinte menu:

// Binary Tree
typedef struct No {
    int data;
    struct No* esquerda;
    struct No* direita;
} Inteiros;

Inteiros* criarNo(int data);
Inteiros* inserirNo(Inteiros* root, int data);
void preOrdem(Inteiros* root);
void ordem(Inteiros* root);
void posOrdem(Inteiros* root);
void liberarArvore(Inteiros* no);


int main(void) {
  Inteiros* root = NULL;
  int opc, num;
  liberarArvore(root); // Limpar

  do {
      printf("\nMenu:\n");
      printf("1 - Inserir nó na árvore\n");
      printf("2 - Percurso em Pré-Ordem\n");
      printf("3 - Percurso em Ordem\n");
      printf("4 - Percurso em Pós-Ordem\n");
      printf("5 - Sair\n");
      printf("Escolha uma opção: ");
      scanf("%d", &opc);

      switch (opc) {
          case 1:
              printf("Digite um número inteiro: ");
              scanf("%d", &num);
              root = inserirNo(root, num);
              break;
          case 2:
              printf("Percurso em Pré-Ordem: ");
              preOrdem(root);
```

```
            printf("\n");
            break;
        case 3:
            printf("Percurso em Ordem: ");
            ordem(root);
            printf("\n");
            break;
        case 4:
            printf("Percurso em Pós-Ordem: ");
            posOrdem(root);
            printf("\n");
            break;
        case 5:
            printf("Encerrando o programa.\n");
            liberarArvore(root); // Limpar
            break;
        default:
            printf("Opção inválida. Tente novamente.\n");
    }
  } while (opc != 5);

  return 0;
}



Inteiros* criarNo(int data) {
    Inteiros* novoNo = (Inteiros*)malloc(sizeof(Inteiros));
    novoNo->data = data;
    novoNo->esquerda = novoNo->direita = NULL;
    return novoNo;
}

Inteiros* inserirNo(Inteiros* root, int data) {
    if (root == NULL) {
        root = criarNo(data);
    } else if (data <= root->data) {
        root->esquerda = inserirNo(root->esquerda, data);
    } else {
        root->direita = inserirNo(root->direita, data);
    }
    return root;
}

void preOrdem(Inteiros* root) {
    if (root == NULL) return;
    printf("%d ", root->data);
    preOrdem(root->esquerda);
```

```
    preOrdem(root->direita);
}

void ordem(Inteiros* root) {
    if (root == NULL) return;
    ordem(root->esquerda);
    printf("%d ", root->data);
    ordem(root->direita);
}

void posOrdem(Inteiros* root) {
    if (root == NULL) return;
    posOrdem(root->esquerda);
    posOrdem(root->direita);
    printf("%d ", root->data);
}

// Função Recursiva
void liberarArvore(Inteiros* nó) {
    if (nó == NULL) {
        return;
    }

    liberarArvore(nó->esquerda);
    liberarArvore(nó->direita);
    free(nó);
}
```