1. Dê a estrutura para uma árvore M-ária de busca em C e uma função que busca uma chave nessa árvore.

```c
#include <stdlib.h>

#define M 10

typedef struct No {
  int chaves[M-1];
  struct No * filhos[M];
} No;

typedef No * p_no;

p_no busca(p_no raiz, int x) {
  int i;
  if (raiz == NULL)
    return NULL;
  for (i = 0; i < M-1 && raiz->chaves[i] < x; i++) ;
  if (i < M - 1 && raiz->chaves[i] == x)
    return raiz;
  return busca(raiz->filhos[i], x);
}
```

2. Dê a estrutura para uma árvore digital de busca onde a chave são strings todas de comprimento K e uma função que busca uma chave nessa árvore.

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

#define K 10

typedef struct No {
  char chave[K];
  struct No *filhos[256];
} No;

typedef No * p_no;

p_no busca_rec(p_no raiz, char x[], int nivel) {
  if (raiz == NULL || strcmp(x, raiz->chave) == 0)
    return raiz;
  return busca_rec(raiz->filhos[x[nivel]], x, nivel + 1);
}

p_no busca(p_no raiz, char x[]) {
  return busca_rec(raiz, x, 0);
}
```

3. Escreva uma função que calcula o número de folhas em uma árvore dada.

```c
int folhas (p_no raiz) {
  if (raiz == NULL)
    return 0;
  if (raiz->esq == NULL && raiz->dir == NULL)
    return 1;
  return folhas(raiz->esq) + folhas(raiz->dir);
```

```
}
```

4. Escreva uma função que compara se duas árvores binárias de busca são iguais.

```
int compara(p_no raiz1, p_no raiz2) {
  if (raiz1 == NULL && raiz2 == NULL)
    return 1;
  if (raiz1 != NULL && raiz2 == NULL)
    return 0;
  if (raiz1 == NULL && raiz2 != NULL)
    return 0;
  /*neste ponto nenhuma eh NULL*/
  return raiz1->chave == raiz2->chave &&
    compara(raiz1->esq, raiz2->esq) &&
    compara(raiz1->dir, raiz2->dir);
}
```

5. Considere a seguinte `struct` usada para representar um nó de uma árvore ordenada

```
typedef struct No {
  int dado;
  No * primeiro_filho, proximo_irmao;
} No;

typedef No * p_no;
```

a) Faça uma função que percorre uma árvore ordenada em pré-ordem.

```
void pre_ordem(p_no raiz) {
  if (raiz) {
    printf("%d\n", raiz->dado);
    pre_ordem(raiz->primerio_filho);
    pre_ordem(raiz->proximo_irmao);
  }
}
```

b) Faça uma função que percorre uma árvore ordenada em pós-ordem.

Similar, basta colocar o printf após as chamadas das funções.