Seznamy v dynamické paměti

Seznam je lineární datová struktura obsahující konečnou posloupnost prvků stejného typu.

Přístup k prvkům seznamu je pouze z jeho začátku – zde lze prvky přidávat a odebírat.

K dalším prvkům za začátkem se přistupuje jen postupně, přes předcházející prvky – seznam je *jednosměrně vázaný*.

Seznamy v dynamické paměti

Typ `Object` je typ prvků, které budeme skládat do seznamu.

```
typedef struct {
    unsigned int id;
    char *name;
} Object;
```

Seznam se skládá z uzlů, které kromě ukazatele na datový prvek obsahují také vazbu na další uzly. Je reprezentován ukazatelem na první uzel.

```
typedef struct node ListNode;
typedef ListNode* List;
struct node {
    Object *data;
    List next;
}
```

Prázdný seznam je reprezentován nulovým ukazatelem.

```
const List listNil = NULL;
```

Přidání uzlu na začátek seznamu.

`px` je ukazatel na prvek, o nějž prodloužíme seznam `*s`.

```
List listCons (Object *px, List *s) {
  List ln = malloc(sizeof(ListNode));
  if (ln != NULL) {
    ln->data = px;
    ln->next = *s;
  }
  *s = ln;
  return ln;
}
```

Napište funkci, která testuje prázdnost seznamu

```
bool listNull (List s)
```

Napište funkce

`Object *listHead (List s)`,
`List listTail (List s)`.

První z nich vrátí ukazatel na prvek v prvním seznamovém uzlu. Pokud je seznam prázdný, funkce vrátí `NULL`.

Druhá funkce vrátí zbytek seznamu bez prvního uzlu. První uzel uvolní z dynamické paměti. Pokud byl seznam prázdný, funkce `listTail` vrátí `NULL`.

Napište funkci

`unsigned int listLength (List s)`,

která vrátí délku seznamu, tj. počet jeho prvků.

Napište funkci

`List listReverse (List s)`,

která podle seznamu `s` vytvoří nový seznam s prvky v opačném pořadí, než byly v seznamu `s`.