```c
/* vim: set tabstop=4 expandtab shiftwidth=4 softtabstop=4: */

#include <assert.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


static void *binary_search(const void *key, const void *base, size_t num_elem, size_t
elem_size, int (*compar)(const void *, const void *));

static int int_cmp(const void *key, const void *elem);

static int str_cmp(const void *key, const void *elem);


int main()
{
    int iary[] = {1, 20, 25, 32, 76, 123};
    int ikey = 76;
    int inokey = 77;
    int *ires = NULL;
    int *ires_check = NULL;
    char *sary[] = {"e01","e02","e03","e04","e05","e06"};
    char *skey = "e01";
    char *snokey = "e07";
    char **sres = NULL;
    char **sres_check = NULL;

    // Case: integer array - key found
    ires = binary_search(&ikey, iary, sizeof iary/sizeof iary[0], sizeof iary[0],
int_cmp);
    ires_check = bsearch(&ikey, iary, sizeof iary/sizeof iary[0], sizeof iary[0],
int_cmp);
    assert( ires == ires_check );
    if (ires != NULL)
    {
        printf("Key %d -> found (element: %d)\n", ikey, *ires);
    }
    else
    {
        printf("Key %d -> not found\n", ikey);
    }

    // Case: integer array - key not found
    ires = binary_search(&inokey, iary, sizeof iary/sizeof iary[0], sizeof iary[0],
int_cmp);
    ires_check = bsearch(&inokey, iary, sizeof iary/sizeof iary[0], sizeof iary[0],
int_cmp);
    assert( ires == ires_check );
    if (ires != NULL)
    {
        printf("Key %d -> found (element: %d)\n", inokey, *ires);
    }
    else
    {
        printf("Key %d -> not found\n", inokey);
    }

    // Case: string array - key found
    sres = binary_search(&skey, sary, sizeof sary/sizeof sary[0], sizeof sary[0],
str_cmp);
    sres_check = bsearch(&skey, sary, sizeof sary/sizeof sary[0], sizeof sary[0],
str_cmp);
    assert( sres == sres_check );
    if (sres != NULL)
    {
```

```
        printf("Key '%s' -> found (element: '%s')\n", skey, *sres);
    }
    else
    {
        printf("Key '%s' -> not found\n", skey);
    }

    // Case: string array - key not found
    sres = binary_search(&snokey, sary, sizeof sary/sizeof sary[0], sizeof sary[0],
str_cmp);
    sres_check = bsearch(&snokey, sary, sizeof sary/sizeof sary[0], sizeof sary[0],
str_cmp);
    assert( sres == sres_check );
    if (sres != NULL)
    {
        printf("Key '%s' -> found (element: '%s')\n", snokey, *sres);
    }
    else
    {
        printf("Key '%s' -> not found\n", snokey);
    }

    return 0;
}

void *binary_search(const void *key, const void *base, size_t num_elem, size_t elem_size,
int (*compar)(const void *, const void *))
{
    const unsigned char *pcb = base;
    size_t lo;
    size_t hi;
    void *ret = NULL;

    assert( key != NULL );
    assert( base != NULL );
    assert( compar != NULL );

    /*
     * Make sure the number of elements is > 0 to avoid underflow problems when
     * initializing the 'hi' variable
     */
    if (num_elem == 0)
    {
        return NULL;
    }

    lo = 0;
    hi = num_elem-1;
    while (lo <= hi && ret == NULL)
    {
        size_t mid = (lo+hi)/2;
        const void *pmid = pcb + mid*elem_size; // -> &base[mid]
        int cmp;

        cmp = compar(key, pmid);
        if (cmp == 0)
        {
            ret = (void*) pmid; // A cast is needed because we are casting from a const
pointer to a non-const pointer.
        }
        else if (cmp < 0)
        {
            hi = mid-1;
        }
        else if (cmp > 0)
        {
            lo = mid+1;
        }
```

```
        /*
         * At the beginning of each iteration, it may happen that both 'lo' and
         * 'hi' are 0, thus resulting in mid == 0.
         * This means that when 'hi' is updated (i.e., hi=mid-1), we incur in an
         * overflow and the function may loop infinitively.
         * To avoid this, we check for the condition mid == 0 and when true we
         * break the loop.
         */
        if (mid == 0)
        {
            break;
        }
    }

    return ret;
}

int int_cmp(const void *pkey, const void* pelem)
{
    const int *pk = pkey;
    const int *pe = pelem;

    assert( pk != NULL );
    assert( pe != NULL );

    return (*pk > *pe) ? 1 : ((*pk < *pe) ? -1 : 0);
}

int str_cmp(const void *pkey, const void *pelem)
{
    const char **pk = (const char**) pkey;
    const char **pe = (const char**) pelem;

    assert( pk != NULL );
    assert( pe != NULL );

    return strcmp(*pk, *pe);
}
```