```c
#ifndef person_record_included
#define person_record_included

/*! \file
 *
 * This file contains three versions of a PersonRecord data structure to
 * illustrate the way C lays out memory for structs, pointers, and arrays.
 *
 */

/**
 * PersonRecord Version 1 has name, id, address, and age fields.  The name and
 * address are char*, the id and age int.
 *
 * Consider the following code segment:
 *                                                                        <pre>
 *     PersonRecordV1 prv1;
 *     prv1.name = "Jane Doe";
 *     prv1.id = 123456789;
 *     prv1.address = "1 Main St."
 *     prv1.age = 25;
 *                                                                        </pre>
 * Here is a picture of the memory layout of variable prv1:
 *
 *     <img src="images/prv1.jpg">
 *
 * Consider now the following code segment, which dynamically allocates a value
 * of type PersonRecordV1:
 *                                                                        <pre>
 *     PersonRecordV1 prv1p = new(PersonRecordV1);
 *     prv1p->name = "Jane Doe";
 *     prv1p->id = 123456789;
 *     prv1p->address = "1 Main St."
 *     prv1p->age = 25;
 *                                                                        </pre>
 * Here is a picture of the memory layout of variable prv1p:
 *
 *     <img src="images/prv1p.jpg">
 *
 */

typedef struct {
    char* name;                     /**< name is a variable-length string */
    int id;                         /**< id is an int */
    char* address;                  /**< address is a variable-length string */
    int age;                        /**< age is an int */
} PersonRecordV1;

/**
 * PersonRecord Version 2 has the same fields as a Person Record Version 1,
 * except the name and address fields are fixed character arrays instead of
 * char*.  These fields are 30 and 50 chars, respectively.
 *
 * Consider the following code segment:
 *                                                                        <pre>
 *     PersonRecordV2 prv2;
 *     strcpy(prv2.name, "Jane Doe");
 *     prv2.id = 123456789;
 *     strcpy(prv2.address, "1 Main St.");
 *     prv2.age = 25;
 *                                                                        </pre>
 * Here is a picture of the memory layout of variable prv1:
 *
 *     <img src="images/prv2.jpg">
 *
 */
typedef struct {
    char name[20];                  /**< name is a string of max 20 chars */
    int id;                         /** id is an int */
    char address[25];               /** address is a string of max 25 chars */
    int age;                        /** age is an int */
} PersonRecordV2;

/**
 * Name is an extended version of the name field in a person record.  It's used
 * in PersonRecord Version 3.  Instead of the the simple char* name field in a
 * PersonRecordV1, the Name struct has three char* fields for first, middle,
 * and last names.
 */
typedef struct {
    char* first;                    /**< first name is a string */
    char middle_initial;            /**< middle initial is just one char */
    char* last;                     /**< last name is a string */
} Name;

/**
 * Name is an extended version of the name field in a person record.  It's used
 * in PersonRecord Version 3.  Instead of the the simple char* address field in
 * a PersonRecordV1, the Address struct has six separate fields for number,
 * street, etc.
 */
typedef struct {
    int number;                     /**< street number is an int */
    char* street;                   /**< street name is a string */
    char* city;                     /**< city name is a string */
    char state[2];                  /**< state is limited to a 2-char string */
    char* country;                  /**< country is a string */
    int zip;
} Address;

/**
 * PersonRecord Version 3 has the same fields as Versions 1 and 2, except here
 * in V3, the name and address fields are structs, no just strings.  The reader
 * is invited to draw a picture.
 */
typedef struct {
    Name name;                      /**< name is defined by the Name type */
    int id;                         /**< id is an int */
    Address address;                /**< address is defined by the Address type */
    int age;
} PersonRecordV3;

```

```
/**
 * Print out a PersonRecordV1.
 */
void printPersonRecordV1(PersonRecordV1 prv1);

/**
 * Print out a PersonRecordV1*.
 */
void printPersonRecordV1p(PersonRecordV1* prv1p);

/**
 * Print out a PersonRecordV2.
 */
void printPersonRecordV2(PersonRecordV2 prv2);

/**
 * Print out a PersonRecordV3.
 */
void printPersonRecordV3(PersonRecordV3 prv3);

#endif
```