Advanced Programming Techniques (a.k.a. Programming in ANSI / ISO C)

Module 8– Dynamic Memory Allocation

Assume the following code:

```
typedef struct person
{
     char *name;
     char *address;
     unsigned int age;
} Person;
```

1. Write a function to allocate memory for a Person and return a pointer to it. Assume *name* should provide a buffer of 20 characters, and *address* of size 30.

2. Does your function in Q1 have any potential memory leaks? Compare your code with your colleagues and discuss.

3. Write a function, *clonePerson* which has a pointer to Person as a parameter, and returns a pointer to a new, duplicate, Person.

4. Assume an *address* should now have a size of 50. Write a function that takes an existing Person and returns a pointer to the same Person with a modified *address* to meet this requirement. Any data in *address* should be maintained. On failure, the function should return NULL.

5. Assume we have a pointer *Person *p* which is currently pointing to a Person. We no longer require this data structure and so perform the following:

   ```
   free(p);
   ```

   What is the effect of this operation? What changes, if any, would you make, and why?