COMP2510 – Review Exercise 4 (Winter 2017)

1. Suppose we have a linked list where each node is dynamically-allocated & contains a record:

```
typedef struct record  record;
struct record {
  char  name[32];
  int   score;
};

typedef struct node  node;
struct node {  /* a node in a linked list */
  record   data;
  node     *next;
};
```

(a) Write a C function `print` with prototype

```
void print(node *lst);
```

that prints all the names & scores (in some suitable format) stored in the linked list `lst` (i.e. `lst` points to the first node of the linked list to print).

(b) Write a C function `find` with prototype

```
node *find(node *lst, const char *name);
```

that searches for a node that contains a name that matches `name` in the linked list `lst`. If there is a match, a pointer to the first node that matches is returned; otherwise, the null pointer is returned.

(c) Write a C function `destroy` with prototype

```
void destroy(node *lst);
```

that deallocates all the nodes in the linked list `lst`.

(d) Assume that `head` points to the very first node of a list. Write a C function `insert` with prototype

```
int insert(node **plst, const char *name, int score);
```

so that the invocation

```
insert(&head, a_name, a_score);
```

creates a new node containing a record with name `a_name` & score `a_score` & inserts that node at the beginning of the linked list. Note that `head` is also updated to point to the new node. The function returns 1 if the operation succeeds; otherwise, it returns 0.

(e) Assume that `head` points to the very first node of a list. Write a C function `delete_all` with prototype

```
size_t delete_all(node **plst, const char *name);
```

so that the invocation

```
delete_all(&head, a_name);
```

deletes all nodes that contain the name `a_name` from the list represented by `head`. It returns the number of nodes deleted.

2. Given the following

```
typedef struct record  record;
struct record {
  char   name[32];
  int    score;
};
```

Suppose we want to sort an array of "records" using qsort. Write a suitable comparison function with prototype: `int cmp(const void *, const void *);` so that

(a) the array is sorted in descending order of scores.

(b) the array is sorted in ascending order of names & if several records have the same name, they are then sorted in descending order of their scores.

3. In each of the following, indicate whether the statement marked (@) is valid. (We regard a statement as invalid if either it won't compile or if it may cause a runtime error.) If the statement is invalid, explain why; if the statement is valid, indicate the output of the printf statement that follows.

(a)
```
char a[]  = "hello";
char *p   = "world";
p = a + 2; /* (@) */
printf("%s", p);
```

(b)
```
char a[]  = "hello";
*a = *(a + 1); /* (@) */
printf("%s", a);
```

(c)
```
char a[]  = "hello";
char *p   = "world";
*(p + 1) = a[0]; /* (@) */
printf("%s", p);
```

4. Describe fully in words what x is in each of the following:

(a) `int (*x)[2];`

(b) `int (*x)(float);`

(c) `int (*x[2])(float);`