COMP1511 18s2

Week-09 Tutorial Exercises

1. The tutorial will start with a code review.
   Discuss the good, the bad and the ugly aspects of their code.

   Please be gentle in any criticism - we are all learning!

2. What is a memory leak?
   What does dcc --leak-check do?

3. Implement a function *list_append* which appends its second argument to its first. It should have this prototype:

   ```
   struct node *list_append(struct node *list1, struct node *list2);
   ```

   As usual, struct node has this definition:

   ```
   struct node {
       struct node *next;
       int          data;
   };
   ```

   It should not create (malloc) any new list elements.
   It should just change the appropriate next field in the first list.

4. Consider:

   ```
   char s[] = "Hello World!";
   char *cp = s;
   char *cp2 = &s[8];
   ```

   What is the output when the following statements are executed?

   ```
   printf("%s\n", cp);
   printf("%c\n", *cp);
   printf("%c\n", cp[6]);
   printf("%s\n",cp2);
   printf("%c\n",*cp2);
   ```

5. Write a function

   ```
   int non_decreasing(int n, int a[n])
   ```

   which checks whether items in an array are sorted in non-decreasing order. (i.e. **a[i] ≥ a[i-1]**, for **0<i<N**). Your function should returns **1** if the items are in non-decreasing order, **0** otherwise.

6. a. Write a function **strings_to_list** which takes an array of pointers to strings and converts it to a linked list. It should have this prototype:

      ```
      struct node *strings_to_list(int len, char *strings[]);
      ```

      Assume the strings contain only digit characters,
      It might be called like this:

      ```
          char *powers[] = {"2", "4", "8", 16"};
          struct node *head = strings_to_list(4, powers);
      ```

   b. How would you use **strings_to_list** to convert a program's command line arguments to a linked list?
   c. How would you use **strings_to_list** to convert a program's command line arguments to two linked lists?
      Assume, a command line argument of "-" separates the arguments to be converted.

Revision questions

The remaining tutorial questions are primarily intended for revision - either this week or later in session.
Your tutor may still choose to cover some of the questions time permitting.

7. Write a function

   ```
   int find_index(int x, int n, int a[n])
   ```

   which takes two integers **x** and **n** together with an array **a**[] of **n** integers and searches for the specified value within the array. Your function should return the smallest index **k** such that **a**[**k**] is equal to **x** (or **-1** if **x** does not occur in the array).

8. We have student fines in a file named fines.txt this format:

```
Linus Torvalds fined $98 for not attending lectures.
Denis Ritchie fined $50 for eating in labs.
Ken Thompson fined $150 for attending lecture in his underpants.
```

   Write a program `student_fine.c` which reads this file and prints the student with biggest fine including the amount and reason in this format.

```
$ a.out
Biggest fine was $150 given to Ken Thompson for 'attending lecture in
 his underpants'.
```

9. Write a function, prototype below, that mirrors the behaviour of the library function **strrchr**. This function takes a string and a character as arguments, and returns a pointer to the last occurrence of the character `c` in the string **s**. It returns **NULL** if the character cannot be found.

```
char *strrchr(char s[], char c)
```

COMP1511 18s2: Programming Fundamentals is brought to you by
the School of Computer Science and Engineering at the University of New South Wales, Sydney.
For all enquiries, please email the class account at cs1511@cse.unsw.edu.au
CRICOS Provider 00098G