CS350 Take Home Exercises

March 6, 2009

These exercises are designed to help you familiarize with the C language. As this class is not about teaching C, these exercises will not be graded.

Problem 1. Write a program that prints "hello world" and a new line to standard output. Your program must include the appropriate header file(s) and come with a main function.

Problem 2. Implement the function strlen() from string.h. This function takes a string and returns the number of characters the string has. A string is a sequence of characters terminated by a zero character, '\0'. If `s` points to a string, then `s[strlen(s)] == '\0'`.

The function has the following prototype.

```
size_t strlen(const char *s);
```

The `size_t` type is an unsigned integer that is large enough to hold the size of any memory object.

Recall that in C, a pointer and an array are interchangeable. In the skeleton below, you can use either `s[i]` or `*(s + i)` to access the i-th character from the string `s`. You can also do `*s` followed by `s++` to reference the pointer and increment it to point to the next character (these two operations can be done in one statement, as `*s++`).

Recall the formula for sample mean and variance: $\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i \qquad S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})^2$

Problem 3. Given an array of doubles and an integer for the array size, compute the sample mean and variance and store the result to a structure.

```
typedef struct {
  double mean, var;
} sample_info_t;

void compute_sample_info(sample_info_t *info, const double *xs, int n);
```

Problem 4. Suppose we have the following definition for a singly-linked list node:

```
typedef struct node_s {
  struct node_s *next;
  int key;
  char *value;
} node_t;
```

This is an associative list, one that stores entries of key-value mappings. Write a lookup function such that, given a key and a list (actually, a pointer to the first node of the list), returns pointer to the node that has the specified key. The function should have this prototype:

```
node_t *assoc_lookup(int key, node_t *head);
```

For the following problem, you may find these functions from ctype.h useful:

```
/* From <ctype.h> */
int isalpha(char c);   /* Returns 1 if c is an alphabet; 0 otherwise. */
char toupper(char c);  /* Converts character c to uppercase. */
char tolower(char c);  /* Converts character c to lowercase. */
```

Problem 5. Write a function that takes a string and decides whether it is a palindrome. A palindrome is a string that reads the same forwards and backwards, permitting relaxation to spaces, punctuations, and case. For example, "Was it a car or a cat I saw?" is a palindrome. "Never odd or even." is another one.

The function should have the following prototype, and returns 1 if the string is a palindrome; 0 otherwise.

```
int is_palindrome(const char *s);
```

You may want to use strlen() from Problem 2.

Problem 6. Write a "tac" program. It reads the standard input line by line and outputs the lines to standard output in reversed order. As the standard input is part of a pipe and therefore not seekable, you will need to store the content in memory.