Introduction To Computer – C (234112)
Winter 2012
Moed B

Duration: 119 minutes. You are not allowed to exit the classroom to use the bathroom.
External Material: you are not allowed to use any written, printed or electronic external material.

Guidelines and Instructions:
- Write your answers only on the exam form, in the intended places. Note that the given place doesn't necessarily indicate the length of the correct answer.
- The even pages of the exam form are empty. You can use them as a draft or to write your answers. Write drafts clearly, so they won't be checked.
- Write your answers tidy, clean and clearly as possible. You're allowed to use pencil and an eraser, but you must fill the title page in pen.
- In all of the questions, you're allowed to define (and implement) your own functions.
- You are not allowed to use global and/or static variables, or pre-compilation commands (include\define).
- You are not allowed to use library functions, or functions implemented in class, without implementing them yourself, unless noted explicitly in the question, excluding input\output functions.
- In every question, you are allowed to use functions defined in previous parts of the same question, even if you didn't solve these parts, though this is not obligated.
- You don't need to check input correction, unless explicitly noted in the question.
- You don't need to implement the main function, and the order of writing the functions is not important.
- the complexity of the solution will not be marked, unless explicitly stated.
- It's recommended to add a written explanation of your algorithm. That explanation will not be marked.
Question 1 (25 Points)
Part 1 (15 Points)
Write the following function:

```c
int shift(int n)

Its parameter is a positive (equal or greater than zero) integer. The function returns the number after shifting the rightmost digit to the leftmost position. If the rightmost digit is 0, then there’s no meaning to putting it in the leftmost position.

Examples:
- For n=3, the function will return 3.
- For n=37, the function will return 73.
- For n=4567, the function will return 7456.
- For n=4560, the function will return 456.

```
Part 2 (10 Points)

Write the following function:

```c
int is_shift(int a, int b)
```

Its parameters are two integers, `a` and `b`. The function returns 1 if after any number of shifts (as described in part 1) on `a`, you can get `b`. If you can't, the function will return 0.

Example:

- For `a=73`, `b=37`, the function will return 1.
- For `a=67`, `b=75`, the function will return 0.
- For `a=675`, `b=670`, the function will return 0.
- For `a=750`, `b=75`, the function will return 1.
- For `1750`, `b=57`, the function will return 1.

```c
int is_shift(int a, int b) {
    int length_a = 1, tmp_a = a, i;

    while(tmp_a/=10) length_a++;
    length_a *= 2;

    for (i=0; i<length_a; i++) {
        if (a == b) return 1;
        a = shift(a);
    }

    return 0;
}
```
Question 2 (45 Points)
Part 1 (5 Points)
Write the following function:

```c
int words_num (char * s)
```

Its parameter is a legal string of characters s (ends with '\0'). The function counts the number of words in the string. The string contains only uppercase and lowercase letters. Between every two words there's exactly one space, and there are no spaces before the first word, or after the last word.

**Example:** for the string "Do not worry be happy", the function will return 5.

```c
int words_num (char * s){
    int i, count = 0;
    if (s[0] == 0)
        return 0;
    for (int i=0; s[i] != '\0'; i++)
    {
        if (s[i] == ' ')
            count++;
    }
    return (count +1);
}
```
**Part 2 (5 Points)**

Write the following function:

```c
int word_idx(char *s, int k)
```

Its parameters are an array of characters (doesn't end with '\0') `s`, and an integer `k`. The function returns the index of the first letter of the `k`th word in the array. The array contains only uppercase and lowercase letters. There is exactly one space between every two words. There are no spaces before the first word. **There is exactly one space after the last word.**

You can assume the input is legal.

**Examples:**

For the array `not worry` and `k=1`, the function will return 0.
For the array `not worry` and `k=2`, the function will return 4.

```c
int word_idx(char *s, int k) {
    int i=0, count = 1;
    while (1) {
        if (s[i] == ' ')
            count++;
        if (count==k)
            return (i+1);
        i++;
    }
    return -1;
}
```

**another solution:**

```c
int i=0, count = 1;
for (i=0; count<k; i++)
    { if (s[i] == ' ')
        count++;
    }
return i;
```
Part 3 (10 Points)
Write the following function:

```c
void rotate_left(char *s, int n)
```

Its parameters are an array of characters (doesn't end with \"\0\") `s`, and its length, `n`. The function shifts the letters in the array one place to the left, so that after the shift, the letter in the first cell will be in the last cell.

**Example:** for the array `not worry`

The function will shift it to `ot worry n`

```c
void rotate_left(char *s, int n){
    int i;
    char tmp = s[0];
    for (i = 0; i < n-1; i++)
        s[i] = s[i+1];
    s[n-1] = tmp;
}
```
Part 4 (10 Points)
Write the following function:

```c
void swap_words(char *s, int n)
```

Its parameters are an array of characters (doesn't end with '\0') `s`, and its length, `n`. the array contains exactly two words with one space between them, and another space after the second word. The function switches the two words. The array contains only uppercase and lowercase letters. There are no spaces before the first word.

**Example:** for the array `not worry`

The function will change it to `worry not`

```c
void swap_words(char *s, int n){
    int i, wi = word_idx(s, 2);
    for (i = 0; i < wi ; i++)
        rotate_left (s, n);
}
```
Part 5 (15 Points)
In this part you can use the function `int strcmp(char *s1, char *s2)`. This function receives two arrays of characters and their lengths, when the last character in both arrays must be a space, and the rest of the array is uppercase and lowercase letters. The function compares the words in lexicographic order. If the first one is bigger, it returns -1. If the second one is bigger, it returns 1. If they're equal, it returns 0.

Write the following function:

```c
void sort_string(char *s)
```

its parameter is a legal string `s` (ends with '\0'). The function uses bubble-sort to sort the words in the string by their lexicographic order, from smallest to biggest. The string contains only uppercase and lowercase letters. Between any two words there's exactly one space, and there are no spaces before the first word, or after the last one. You can use functions from previous parts even if you didn’t implement them.

Example: for the string "do Not worry be happy", the function will change it to "Not be do happy worry".

```c
void sort_string(char *s) {
    int end_word2=0, i=0, j=0, slen=0, n = words_num(s);

    /*calculating total string length*/
    while(s[slen] != '\0')
        slen++;

    s[slen]=' '; /*otherwise swap_words will not work on last word*/
    for (i=n; i>1; i--){
        for (j=1; j<i; j++){
            if (strcmp(s+word_inx(s,j), s+word_inx(s,j+1)) < 0){
                end_word2 = (j==n-1) ? slen+1 : word_inx(s,j+2); /*calculating where second word ends + 1*/
                swap_words(s+word_inx(s,j), end_word2-word_inx(s,j));       
            }
        }
    }
    s[slen] = '\0';
}
```
**Question 3 (30 Points)**

**Part 1 (15 Points)**

A "prefix" of a string is a sequence of adjacent letters in the beginning of the string. **Example**: for the string "boy", its prefixes are "boy", "bo", "b", and "" (empty preface).

Write the following function:

```c
int prefix(char a[], char b[]){

Its parameters are two legal strings, a and b (end with '\0'). The function returns the length of the longest shard prefix. Don't include the termination character ('\0') in your calculations.

**Examples:**
- For the strings "Hello", "Hello world", the function will return 5.
- For the strings "Hello", "World", the function will return 0.

**You are not allowed to use recursion at all.**

```c
int prefix(char a[], char b[]){
    int counter;
    for(counter=0 ; (a[counter] == b[counter]) && (a[counter] != 0) ; counter++);
    return counter;
}
```
Part 2 (15 Points)

A "prefix" of a string is a sequence of adjacent letters in the beginning of the string.
Example: for the string "boy", its prefixes are "boy", "bo", "b", and "" (empty preface).

Write the following recursive function:

```c
int rec_prefix(char a[], char b[]){

Its parameters are two legal strings, a and b (end with '\0'). The function returns the length of the longest shard prefix. Don't include the termination character ('\0') in your calculations.

Examples:
• For the strings "Hello", "Hello world", the function will return 5.
• For the strings "Hello", "World", the function will return 0.

You are not allowed to use loops at all.

int rec_prefix(char a[], char b[]){
    if(*a != *b || (*a == 0))
        return 0;
    return 1 + rec_prefix(a+1, b+1);
}
```