CS 2213-002 Data Structures

Spring 2014 – Midterm1 -- Feb 27, 2014
You have 75 min. Good luck.
You can use the 2-page C reference card posted in the class web page.

Name:…… Score: ……./100

Background Survey (10pt bonus credit)

A. Please complete the below table for the computer-programming-related courses that you have taken before Spring 2014.

Programming courses	TAKEN AT UTSA, (Yes / No)	If you have taken equivalent courses from a **different** school, please give the **school name** and the **programming language** used.
CS 1063 Intro to Comp Prog I (in Java)		
CS 1713 Intro to Comp Prog II (in C)		
CS 2123 Data Structures (in C)		

B. How would you evaluate your programming skills and background? (circle one)

5: excellent, contributed to commercial or open-source software.
5: excellent, received A in all Intro programming courses.

4: good, have not take any programming course, but wrote programs more than 1000 lines.
4: good, received A-B in Intro programming courses.

3: fair, have not take any programming course, but wrote programs less than 1000 lines.
3: fair, received B-C in Intro programming courses

2: need improvement, received C-D in Intro programming courses.

1: beginner, have not take any programming course, never wrote any program.
0: What is "programming"?

C. Do you think you knew the background subjects at the level that I reviewed so far?

○ Definitely. ○ Somewhat. ○ I don't know. ○ Not really. ○ Definitely not.

D. Do you think NOW you have sufficient background to take the rest of CS 2123 or need more practice on the background subjects that we covered so far? (Please comment)

Name:.....

1. (20 pt) You are asked to implement the following function using pointers and pointer arithmetic

```
int substrindex(char *str, char *substr);
```

which returns `-1` if `substr` is not in `str`; otherwise, it returns the index value for the first appearance of `substr` in `str`. You can assume that both `str` and `substr` are NULL terminated strings. For example,

`substrindex("abcabcdef", "abc")` returns `0`,

`substrindex("abcabcdef", "abcde")` returns `3`,

`substrindex("aaaab", "aaab")` returns `1`,

`substrindex("abc", "xy")` returns `-1`

```
int substrindex(char *str, char *substr)
{
   /* FOR FULL CREDIT, USE POINTER and POINTER ARITH NOTATION */
   /* IF YOU USE ARRAY NOTATION, YOUR MAX WILL BE 15pt */
```

Name:.....

Name.....

2. (20 pt) A **word search** is a game where letters of words are hidden in a grid (2D char array), that usually has a rectangular or square shape. The objective of this game is to find and mark all the words hidden inside the grid. The words may be hidden horizontally (left-to-right →), vertically (top-to-bottom ↓) or diagonally (left-top-to-right-bottom ↘). Actually you solved this problem for the three cases in HW2.

   Now you are asked to write a function that implements **reverse horizontal** (right-to-left ←) search to find out if a given word (a null terminated string) appears **reverse horizontally** in a given grid (2D array of characters, rows or columns are NOT null terminated). If the word appears reverse horizontally in the grid then your function should return 1 as well as the index values for the beginning row and column numbers (br, bc) of the hidden word in the grid; otherwise, the function returns 0.

   Here is an example showing how the function that you will implement in the next page might be used in main( ).

```
/* suppose std C libs are included here */

#define ROW 2
#define COL 4      /* these numbers will be large in an actual program */

main()
{
  char g[ROW][COL] = {{'a','b','c','d'},
                      {'x','y','z','d'}};
  char w[128];
  int res, br, bc;

  printf("Enter the word you want to search in the grid : ");
  fgets(w, 127, stdin);

  res = reverse_horizantal( g, w, &br, &bc);   /* YOU WILL IMPLEMENT */
  if(res==1)
     printf("Word %s appears reverse horizontally at (%d, %d)\n", w, br, bc);
  else
     printf("Word %s is not found \n", w);
}
/*
For example when w is "dzy" the program should print
dzy appears reverse horizontally at (1, 3)
*/
```

   Hint: Suppose the function `int substrindex(char *str, char *substr);` that you implemented in Question 1 works correctly and available for you to use in this question. Recall that it returns `-1` if `substr` is not in `str`; otherwise, it returns an index value showing where `substr` starts in `str`. This function expects both `str` and `substr` to be null terminated strings.

Name:.....

```
int reverse_horizantal(char g[][COL], char *w, int *br, int *bc )
{
```

Name:_____

3. (20 pt) **Trace** the following program, **show how values change** in memory, and **give the output.**

```
#include <stdio.h>
main()
{
  int x=3, y=9, z[4]={0}, *p1, **p2;

  p1 = &z[2];

  p2 = &p1;

  *p1++ = 5;

  *(*p2-2) = 8;

  (*--p1)++;

  printf("%d %d %d \n",
         p1, *p1, &p1);

  printf("%d %d %d %d \n",
         p2, *p2, **p2, &p2);

  y = f1(&x, **p2, p1+2, p2);

  printf("%d %d %d %d %d %d\n",
           x, y, z[1], z[2], p1, p2 );
  printf("%d %p %d\n", *p1, *p2, **p2);

}
int f1(int *a, int b, int *c, int **d)
{
   int x=5, y=13;

   *a = y / 2 % x;

   *d = c-1;

   **d = *(*d-1) + b;

   return *a * **d;
}
```

MEMORY

name	Address	Content/Value
x	12	
y	16	
z[0]	20	
z[1]	24	
z[2]	28	
z[3]	32	
p1	36	
p2	40	
	100	
a	104	
b	108	
c	112	
d	116	
x	120	
y	124	
	128	

Name:.....

4. (20pt) Write a program that takes the name of a C source file as a command line argument (e.g., `> prog source.c`) and then prints out only the comments in that C file on the screen (i.e., everything between `/*` and `*/` in that C file). You don't need to consider `//` comments.

```
#include <stdio.h>
#include <stdlib.h>

void main(.....)
{
```

Name:.....

5. (20pt) Suppose we have the following structure (record) declarations and the main program, which uses two functions that you will be asked to implement in the next page.

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
   int x;
   int y;
} myDataT;

main()
{
     myDataT   **a;
     int row, col;

     printf("Enter Numbers of Row and Col :");
     scanf("%d %d", &row, &col);

     a = Allocate2D( row, col );        /* YOU WILL IMPLEMENT */

          /* suppose we initialize a and
               do some other operations on a[i][j]'s */

     Free2D( a, row );                  /* YOU WILL IMPLEMENT */

}
```

Name:.....

a. (15 pt) Give the implementation of `Allocate2D` used in the above main program. It allocates a 2D array of `myDataT` records and returns the poner to that 2D array. If there is not enough memory, it returns NULL (if this is the case, make sure you will free partially allocated parts before returning NULL).

```
myDataT **Allocate2D(int row, int col)
{
```

b. (5 pt) Give the implementation for `Free2D` used in above main program to release (free) all the memory allocated by the above code.

```
void Free2D(myDataT **a, int row)
{
```