香 港 中 文 大 學
The Chinese University of Hong Kong

Course Examination 1st Term, 2014 – 2015

Course Code & Title : ESTR1002 Problem Solving by Programming

Time allowed : 2 hours

Student I.D. No. : _____ Seat No. : _____

- Answer **ALL** Questions. Full Score is 100.
- Please mark all your answers on the answer book provided.
- On the answer book, use a new page for each problem.
- List of C operators and the ASCII table are listed on the last page of this question papers.
- For questions that require you to show the output, use the symbol ␣ to denote space in your answer where needed.
- For all problems, you may assume **int** is 4 bytes.
- For all problems, you may assume all input values (from keyboard or from file) are valid.

Problem 1: [2% x 10 = 20%]

For each of the following segments of code, show the output produced by the code.

a.
```
char s1[10] = "\\\n";
char s2[10] = { 'A', '\0', 'B' };
printf("%d %d", strlen(s1), strlen(s2));
```

b.
```
int a = 10;
int *p1 = &a;
printf( "%d\n", (a = a + 1) );
printf( "%d\n", *p);
```

c.
```
printf("%c %d", 'B'-'A'+'a', '1' + 1);
/* note: ASCII code there is on the last page */
```

d.
```
char s1[10] = "ABCD";
int i = 0;
while ( s1[i] != '\0') i++;
printf( "%d", i );
```

e.
```
printf("%d ", 5 / 2 < 2 * 4 / 3);
printf("%d ", 1 + 1 || 1 && 0);
/* note: operator precedence table is on the last page */
```

f.
```
double x = -3 / 4;
int y = 3 / 4.0;
printf("%.2lf %d", x, y);
```

g.
```
int A[100] = { 1, 2, 3, 4, 5 };
int B[] = { 1, 2, 3, 4, 5, 0};
printf("%d %d", A[B[1]], A[5] == B[5]);
```

h.

```
int a = 3, b = 7;
if (a / b)
printf("A\n");
printf("B\n");
if (a / 2)
printf("C\n");
else
printf("D\n");
```

i.

```
int foo(int x, int y) {
    if( x == 0)
        return 0;
    return y + foo(x-1, y);
}

int main(void) {
    printf("%d", foo(20, 3));
    return 0;
}
```

j.

```
int count = 0, x = 5, y = 2;

if (x % 2) {
    int count = 0;
    count++;

    if (y / 2) {
        int count = 2;
        count *= 100;
    }

    printf("%d\n", count);
}

printf("%d\n", count);
```

Problem 2: If-Else [15%] (Common Question)

Write a segment of code to ask the user for two integers, *M* and *N*, and then print the value that is closer to 10. If the difference between *M* and 10 is the same as the difference between *N* and 10, your program should print the larger value. You can assume *M* and *N* are distinct integers.

Sample Output 1 (underlined characters are input entered by the user)
8 13 **8**
Sample Output 2 (underlined characters are input entered by the user)
7 13 **13**

Problem 3: 2D Array [10%] (Common Question)

a) [2%] Declare a 2D array named **`field`** for storing characters (values of type **`char`**).
The first dimension of the array should be 12, and the second dimension of the array should be 7.

b) [1%] How much memory space (in bytes) is required to store the above array in the memory?

c) [3%] Write a segment of code to set ALL elements of the above array to `'#'`.

d) [4%] Write a segment of code to set all the border elements (top row, bottom row, left column and right column) to `'@'`.

Problem 4: Nested Loop [15%] (Common Question)

Write a segment of code to read an integer from the user and then print a triangle as shown in the sample output. You can assume the input value is an integer no less than one.

Sample Output 1 (underlined characters are input entered by the user)
5 ***** **** *** ** *
Sample Output 2 (underlined characters are input entered by the user)
3 *** ** *

Problem 5: Function [10%] (Common Question)

a) [7%] Complete the implementation of the following function:

```
#include <math.h>

/* This function calculates √number and number^4 in one kick.
   The TWO results are passed-back-by-address without printing.

   If "number" is non-negative, the function returns 0.
   If "number" is negative, the function returns 1. In addition,
   the value pointed by ptr_square_root should not be modified.
*/

int calculate(double   number,           /* INPUT  parameter */
              double * ptr_square_root,  /* OUTPUT parameter */
              double * ptr_power_four)   /* OUTPUT parameter */
{
    ... // This is the part you need to write
}
```

Note: The prototype of the function for computing square root is: **`double sqrt(double x)`**

b) [3%] Write a segment of code to call the function, `calculate()`, defined in (a) to compute the square root of **2357.8642** and **2357.8642**^{4}.

In this part,
- you may assume the function in (a) has been correctly implemented,
- you may declare additional variables as you see fit, and
- you are given the following main function to start your implementation and you MUST use only `calculate()` to compute the results.

- Last, print the results rounded to 4 decimal places.

```
int main(void) {
    return 0;
}
```

Problem 6: From Recursion to Iteration [10%]

Given the following recursive function:

```
int f(int n) {
  if( n == 0 )
    return 0;
  else if ( n == 1 )
    return 1;
  else
    return f(n-1) + f(n-2);
```

Write another function satisfying the following requirements:

- carrying the same name and the same list of argument;
- return the same set of output for $0 \leq n \leq 20$;
- **without using recursive function calls**.

Problem 7: Recursion, Strings, and Sorting [20%]

In the lecture, we have introduced the following program:

```
void f (int slot, char alphabet[], int a_len,
                  char result[],   int r_len) {
    int i;
    if (slot == 0) {
        printf("%s\n", result);
        return;
    }

    for (i = 0; i < a_len; i++) {
        result[r_len-slot] = alphabet[i];
        f(slot-1, alphabet, a_len, result, r_len);
    }
}
int main (void) {
    char alphabet[3] = {'9', '6', '8'};
    char result[4]; // 3 slots + '\0'

    result[3] = '\0';
    f(3, alphabet, 3, result, 3);
    return 0;
}
```

a) [3%] What would be the kind of outputs produced by the above program? Choose one from the following 4 options: Permutation with repetition / permutation without repetition / combination / a power set of the alphabets?

Also, show the first 5 lines of output from the program.

b) [7%] Given the following new set of codes:

```
char string_store[?][?];  /* new global 2D char array */

int main (void) {
    // You may add new local variables //
    char alphabet[3] = {'9', '6', '8'};
    char result[4]; // 3 slots + '\0'

    result[3] = '\0';
    f(3, alphabet, 3, result, 3);  // you may add new arguments
    return 0;
}
```

Our goal is to, instead of printing the strings to the screen, store the strings into the global variable "**`string_store`**".

Your task is to:

- Set the **minimum dimensions** for the 2D array "`string_store`" such that it can store all the strings produced by the original function "f". Hint: "`string_store` " is actually an array of strings.
- By modifying the function "f", store all the produced strings, which are originally printed to the screen, into the variable "`string_store`".
- You are allowed to add more arguments to the function "f". Also, you have to update the statement in the main function which invokes the function "f".
- You can add local variables to the main function and the function "f".
- You **cannot** add any extra global variables.
- The order of the strings stored in "`string_store`" is not important.
- You may find the function "`strcpy`" useful:

```
strcpy(a, b);  // copy string b to string a.
```

c) [10%] The last task is to sort the strings stored inside "`string_store`". The rules are given as follows:

- Write a new function as follows:

```
void sort_strings( int num ) { ..... }
```

- The "`sort_strings`" function sorts the global 2D array "`string_store`". The parameter "`int num`" is the number of strings in "`string_store`".
- The main function calls "`sort_strings`" after calling the function "f", and you are not required to write down the calling statement in the main function.
- After storing, `atoi(string_store[i])` must be greater than or equal to `atoi(string_store[i+1])`, for each i from 0 to "`num-2`".
- You cannot assume any order inside the "`string_store`". That means your sorting function cannot assume that the 2D array is already sorted according to the answer of Part (b).
- No codes of any sorting algorithms would be given. Although the lecturer has taught you the selection sort algorithm, you can implement other sorting algorithms as long as what you have written is correct.
- Last but not least, if you insist in reading the entire question, you can find the greeting message from Dr. WONG: "*Wish you a merry Christmas*!"

– END OF PAPER –

.../Attachment

Appendix

List of Partial C Operators in Decreasing Precedence	Associativity
() [] . -> ++ (postfix) -- (postfix)	left-to-right
+ (unary) - (unary) ++ (prefix) -- (prefix) !	right-to-left
* / %	left-to-right
+ (addition) - (subtraction)	left-to-right
< <= > >=	left-to-right
== !=	left-to-right
&	left-to-right
&&	left-to-right
||	left-to-right
= += -= *= /= etc.	right-to-left
, (comma operator)	left-to-right

ASCII Code Table

0 NUL	1 SOH	2 STX	3 ETX	4 EOT	5 ENQ	6 ACK	7 BEL
8 BS	9 HT	10 NL	11 VT	12 NP	13 CR	14 SO	15 SI
16 DLE	17 DC1	18 DC2	19 DC3	20 DC4	21 NAK	22 SYN	23 ETB
24 CAN	25 EM	26 SUB	27 ESC	28 FS	29 GS	30 RS	31 US
32 SP	33 !	34 "	35 #	36 $	37 %	38 &	39 '
40 (	41)	42 *	43 +	44 ,	45 -	46 .	47 /
48 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7
56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G
72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	91 [	92 \	93]	94 ^	95 _
96 `	97 a	98 b	99 c	100 d	101 e	102 f	103 g
104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w
120 x	121 y	122 z	123 {	124 |	125 }	126 ~	127 DEL

****