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

Course Examination 1st Term, 2021 – 2022

Course Code & Title : ESTR1002 Problem Solving by Programming

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

請勿攜去
Not to be taken away

<< The Cover Page >>

Please read the following instructions carefully.

- You are required to answer **ALL** questions. Full Score is 100. **Time allowed is 2 hours**.
- Please write all your answers in **the space provided** on this question paper.
- **If there is not enough space, you may write answers on over-leaf.**
- Please write neatly using pen or pencil.
- Cross-out unwanted writings, otherwise, those may be considered as part of your answers.
- A list of C operators as well as the ASCII table are provided at the end of the paper.
- Unless a *complete* C program is demanded, you may assume inclusion of header files.
- 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.
- Unless otherwise specified, all input/output are console input/output (i.e., keyboard input and screen output).
- In the questions, the notation, `xyz[]`, is used to refer to an array named `xyz`.

Marker's Use Only

Problem	1	2	3	4	5	6	7	Total
Full Score	30	11	15	12	14	8	10	100
Score								

Problem 1: Short Questions [2% each box; 30% total]

Write clearly the output produced by each of the following segments of code.

Problem 1		Answers

a.	`double a = 2.1;`
	`printf("%d\n", (int) (a + 0.5) + (int)0.1);`
	`printf("%.3f\n", a);`	

b.	`int x = 2, y = 3;`
	
	`printf("%d\n", y > x > 1);`
	`if( 1 < x && y%x == 0 )`
	`  printf("%d\n", y == y != y);`
	`else`
	`  printf("%d\n", x >= x >= x);`	

c.	`int B[] = {0, 2, 4, 6, 8};`
	`printf("%d\n", B[3] + B[1]);`
	`B[4] += B[2];`
	`printf("%d\n", B[ B[2] - B[0] ]);`	

d.	`int x = 3, y = 8, tmp;`
	`if (x < y) {`
	`  tmp = x;`
	`  x = y;`
	`  y = tmp;`
	`}`
	`printf("%d\n", x);`
	`printf("%d\n", y);`	

e.	`int x1 = 0, y1 = 3;`
	`int x2 = 2, y2 = 6;`
	
	`printf("%d\n",  x1 * x2 + y2 * y1);`
	`printf("%d\n", (y1-y2)/(x1-x2));`	

Problem 1 cont'd.		Answers

f.	`int x, z, sum = 97;`
	`for (x = -38; x < 49; x++)`
	`{`
	`  if (x > 47)`
	`  printf("%c\n", x);`
	`  else`
	`  for (z = 10; z >= -50; z--)`
	`  if (x == -z)`
	`  sum += x + z;`
	`}`
	`printf("%c\n", sum);`	

g.	`void foo(int i, int sum) {`
	`  sum *= i;`
	`  i -= 2;`
	`}`
	
	`int main() {`
	`  int i, sum = 2;`
	`  for ( i = 9; i > 1; i -= 3 )`
	`  foo(i, sum);`
	
	`  printf("%d\n", sum);`
	`  printf("%d\n", i);`
	`  return 0 ;`
	`}`	

h.	`void bar(int A[], int B) {`
	`  int i;`
	`  for(i=0; i<5; i++)`
	`  A[i] = i * i;`
	`  B--;`
	`}`
	
	`int main(void) {`
	`  int A[5] = { 10 }, B[] = { 7, 6, 5, 4, 3 };`
	`  bar(B, A[2]);`
	`  printf("%d\n", A[2]);`
	`  printf("%d\n", B[2]);`
	`  return 0;`
	`}`	

Problem 1 cont'd.		Answers

i	`char name[9] =`
	`{'E','N','G','G','1','1','1','\0'};`
	`printf("%d\n", name[3] - name[0]);`
	`printf("%c\n", strlen(name));`	

j	`printf("%d\n", strcmp("b", "B") > 0);` `printf("%d\n", strcmp("A", "AA") > 0);`	

k.	`int recursion(int x) {`
	`  int p = x;`
	`  if (x > 1) {`
	`  p = recursion(x-1) * -x;`
	`  }`
	`  return p;`
	`}`
	`int main(void) {`
	`  printf("%d\n", recursion( 2 ) );`
	`  printf("%d\n", recursion( 3 ) );`
	`  return 0;`
	`}`	

l.	`char url[] = "MyCUHK";`
	`int i = 2;`
	`char *s = &url[strlen(url)];`
	`printf("%d\n", url[i]);`
	`printf("%d\n", *s);`	

Problem 1 cont'd.		Answers

m.	`int i = 2, j = 3, k = 4;`
	`int *p = &i, *q = &j;`
	`*p = k;`
	`j = i;`
	`printf("%d\n", i);`
	`printf("%d\n", j - *q);`	

n.	`void f2(int *a, int *b) {`
	`  int tmp = *a + 2;`
	`  *a = b;`
	`  b = tmp;`
	`}`
	
	`int main(void) {`
	`  int x = 8, y = 4;`
	
	`  f2(&y, x);`
	`  printf("%d\n", x);`
	`  printf("%d\n", y);`
	`  return 0;`
	`}`	

Please fill in the blank below.

o.	PBO refers to _____ of _____ Ordinance.

Problem 2: Character and String Processing [11%]

A vowel is one of these five characters: 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O' and 'U'.

i) Write a function `is_vowel( char letter )` that returns an integer value of 1 if the parameter `letter` is a vowel; and returns 0 otherwise. (4%)

ii) Complete the program to read ONE line of text from a user and output the same text but with an extra hyphen (`'-'`) added after each vowel. Your program must read the line of text from the *standard input* (keyboard) using `fgets()`. You may assume the input text contains no more than 80 characters, including the newline character. You can call the function defined in part (i). (7%)

User Input	Sample Output
Good melody	Go-o-d me-lo-dy

User Input	Sample Output
A cat and an ant	A- ca-t a-nd a-n a-nt

```
#include <stdio.h>
int main() {
  // Answer:









}
```

Problem 3: Arrays and Selection Sort [15%]

Complete a program to calculate the **median** of ***n integers*** *(1 ≤ **n** ≤ 100)* obtained from user input. First input is ***n***, followed by ***n integers*** *that may be presented in any order*, so we will use sorting.
If ***n*** is **odd**, the median is the number in the middle, and output it as an **integer**.
If ***n*** is **even**, the median is the average of the two numbers in the middle, and output it as a real number with **one decimal place**.

Sample run 1:

```
5
1 2 1 3 4
2
```

Sample run 2:

```
6
0 3 2 1 0 4
1.5
```

Sample run 3:

```
2
5 5
5.0
```

```
#include <stdio.h>
int main() {
  int i, j, n ;
  int A[ 100 ] ;

  // a) Complete the code to firstly read an integer n (1 ≤ n ≤ 100);
  //    then let the user enter n integers and store them in array A
  // Answer:




  // b) Fill in the blanks to apply selection sort to array A:
  //    to arrange the n numbers in non-descending order
  int minPos , tmp ;
  for ( i = 0 ; ____________________________ ; i++ ) {

    minPos = i ;            // store i first, let it be the smallest

    for ( ____________________________ ; j++ ) {

        if ( ____________________________ )
            minPos = j ;
        if ( minPos != i ) {




        }
    }
  } // end of outer for
```

Problem 3 cont'd.

```
  // c) Complete the program to output the median of the n integers.
  Answer:
```

```
    return 0;
} // end of main()
```

Problem 4: File Processing [12%]

Consider a grade-sheet file that stores student's score data in the following text file format:

File Format	Sample File Content
M SID_1 $ProjectScore_1$ $ExamScore_1$ SID_2 $ProjectScore_2$ $ExamScore_2$ ... SID_M $ProjectScore_M$ $ExamScore_M$	4 1155123456 89 90 1155987654 10 20 1155134679 30 99 1155846213 50 50

- **M** is a positive integer that represents the number of records such that $1 \leq M \leq 200$.
- Then, each of the next **M** lines consists of a set of **SID**, **ProjectScore** and **ExamScore**.
  - Each **SID** is a 10-digit number containing NO alphabets, with **1155** as prefix.
  - Each **ProjectScore** and each **ExamScore** is an integer in range [0,100], and the data values are space-separated.

Please complete the following C program.

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

int main(void) {

    FILE * fptr ;

    // "Line A": Your code starts here

    ...

    // "Line B": Your code ends here

    // Please see below for the requirements

    return 0;
}
```

Your program should read the grade-sheet data stored in a file named **ScoreTable.txt** in the format stated above. If the file cannot be opened successfully, your program should terminate immediately. However, if the file can be opened, you can assume that the file format must be correct.

Consider the total score is calculated as a weighted sum of **ProjectScore** and **ExamScore** :

$$Total = 40\% \times ProjectScore + 60\% \times ExamScore$$

Your program should read each line in the data file and finally print out the **highest** and **lowest total scores** (one decimal point) together with the associated **SIDs** as follows:

A sample run is shown as follows:

Inside "ScoreTable.txt"	Output from your program
4 1155123456 100 100 1155987654 80 80 1155134679 90 99 1155846213 99 80	Max: 1155123456 100.0 Min: 1155987654 80.0

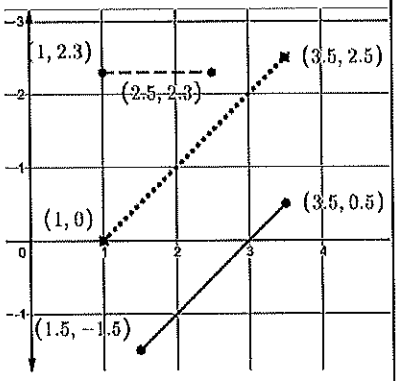
Problem 4 cont'd.

Please provide your answer below. You may write code between `"Line A"` and `"Line B"` above.

Problem 5: Structure and Function [14%]

Consider the following type definitions for representing a 2D line:

```
typedef struct {
  // 2 points (4 coordinates) in a line
  double   x1, y1, x2, y2;
} Line;
```



In this code, you may use functions defined in `<math.h>` and assume `#include <math.h>`.

a) **Write code below** to declare TWO lines, **`line1`** and **`line2`**, of type `Line`, and initialize each of them with the following elements accordingly:
- the *FIRST* line with *all zeroes*, and
- the *SECOND* line with `(1.0, 0.0), (3.5, 2.5)`. (2%)

b) **Write code below** to ask user inputs for **`line1`**. Sample Run: (2%)

```
Input line1: 1.0 2.3 2.5 2.3
```

c) Write a function **`double computeSlope(Line L)`** below to compute the slope of a line. You can assume that the line is valid, i.e., it has non-zero length and it is not vertical. The slope of a line L consisting of endpoints (x_1, y_1) and (x_2, y_2) is given by

$$Slope_L = (y_1 - y_2) / (x_1 - x_2).$$

(2%)

Problem 5 cont'd.

d) Write a function **`void checkParallel(Line *line1, Line *line2)`** to **print** the elements of two **`Line`** parameters and **check** if the two lines are in parallel (coincident lines are considered as parallel as well). Two lines are said to be in parallel if the difference between their slope values is very close to zero, e.g., *line1* and *line2* are in parallel if (8%)

$$-0.001 < (Slope_{line1} - Slope_{line2}) < 0.001$$

You can assume that both lines have non-zero length and are not vertical, i.e., valid slope. Also, all double floating numbers should be output with 1 decimal place.

Sample Run 1:

```
line1: (1.0, 2.3), (2.5, 2.3)
line2: (1.0, 0.0), (3.5, 2.5)
not parallel
```

Sample Run 2:

```
line1: (1.0, 0.0), (3.5, 2.5)
line2: (1.5, -1.5), (3.5, 0.5)
parallel
```

Answer:

Problem 6: Shuffle numbers [8%]

The program below should shuffle TEN integers and print a different sequence each time you run it.

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

void swap( int * ptr1 , int * ptr2 ) {
    int * tmp = ptr1 ; ptr1 = ptr2 ; ptr2 = tmp ;
}

void shuffle( const int * data , int n ) {
    int i , j ;
    for ( i = n-1 ; i > 0 ; i-- ) {
        j = rand() % n ;
        swap( data + i , &( data[j] ) );
    }
}

int main() {
    int i , data[10] = { 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 };
    srand( 1002 );
    shuffle( data , 10 );
    printf("results:");
    for ( i = 0 ; i < 10 ; i ++ )
        printf( " %d" , data[i] );
    printf("\n");
    return 0;
}
```

However, the above program has several bugs. Please find each of them and provide a line of code below for fixing each bug.

Problem 7: How many rounds has Peter played the game? [10%]

In a phone APP, a player can play a mini game as many rounds as he/she wants. One gets **5 points for winning** a round in the mini game, **1 point for a draw**, and **zero points for a loss**. If Peter says that he got 4 points by playing the mini game 3 rounds. Is he telling the truth?

Denote S as the score and N as the number of rounds, write a program to test if it is really possible for Peter to get S points by playing N rounds. For examples:

- S = 3 and N = 4 is possible;
- S = 4 and N = 3 is impossible;
- S = 13 and N = 5 is impossible;
- S = 14 and N = 5 is impossible;
- S = 100 and N = 100 is possible; and
- S = 500 and N = 99 is impossible.

Below is an incomplete program. You may assume the user inputs, $1 \leq N \leq 100$ and $0 \leq S \leq 500$.

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

int test( int S , int N )
{
    // MUST use recursion here
}

int main()
{
    int S, N, result;

    printf("Enter S: \n"); scanf("%d",&S);
    printf("Enter N: \n"); scanf("%d",&N);

    result = test( S , N );  // return either 0 or 1

    if ( result == 1 )
        printf("possible.");
    else
        printf("not possible.");

    return 0;
}
```

First, you **MUST use recursion** to solve the problem (6%). Second, you need to **use memorization** to speed up the computation (4%), so you may modify anywhere of the program.

Please write your code on the next page.

Problem 7 cont'd.

Please provide your answer below. If you want to change `main()`, you do not need to write the entire `main()` again; you just need to write down your code and specify where to put it.

<< Appendix >>

Partial List of C Operators in Decreasing Precedence	Associativity
() [] . -> ++ (postfix) -- (postfix)	left-to-right
+ (unary) - (unary) ++ (prefix) -- (prefix) ! * (unary)	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 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

<< END OF PAPER >>