

1111 NTOUCSE 程式設計 1C Midterm Exam

姓名：_____ 系級：_____ 學號：_____

111/11/01 (二)

考試時間：**13:20 – 16:00**

- Exam rules :
1. **Close book, close** everything including quizzes, homeworks, assignments, reference materials, etc.
 2. You can answer the questions in **English** or in **Chinese**, in this **problem sheet**, the **answer sheet**, or **both**.
 3. You can use language features not taught in class if you feel necessary, but strictly limited in C or C++.
 4. **No mobile phone, pad, computer** or **calculator** is allowed. (Electronic) English dictionary is OK
 4. No peeping around! No discussion! No exchange of any material; **raise your hand if you have any question about the exam problems**
 5. If you turn in the paper earlier than the specified time, **leave the classroom immediately and quietly**
 6. Against any of the above rules will be treated as cheating in the exam and handled by school regulations.
 7. Turn in **BOTH** the **problem sheet with your name and id** and **answer sheet with your name and id**.

1. [10] The following program read several (larger or equal to 3) distinct positive integers with value less than 2^{31} . It expects to print out the 3rd largest number. We have covered a program that finds out the largest in a list of numbers in which we have a variable to keep the maximal value out of those numbers ever been read in to the program. In the case of finding the 3rd largest number, the following program uses three variables to keep the maximal value up to now (max[0]), the 2nd largest value up to now (max[1]), and the 3rd largest value up to now (max[2]). Please finish the following program (You can write your answers on the answer sheet with the line number of the following program or you can put your answers directly on the problem sheet and make a note on the answer sheet)

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int x, max[3]={-1,-1,-1};
06     while (1==scanf("%d", &x))
07     {
08         if (x>max[0])
09         {
10             max[2]=_____;
11             max[1]=_____;
12             max[0]=x;
13         }
14         else if (x>_____)
15         {
16             _____;
17             _____;
18         }
19         else if (x>max[2])
20             max[2]=x;
21     }
22     printf("%d\n", max[2]);
23     return 0;
24 }
```

2. Please read the program below and answer the following questions:

- (a) Which variables are global variables? [3] Which variables are local variables? [2]
- (b) [5] Using this program as example to explain the benefits a programmer might get in using these global variables?

- (c) [5] Using this program as example to explain the troubles a programmer might get into in using these global variables?
- (d) [5] Please use function parameters to modify this program such that the usage of global variables is avoided.

```

01 #include <stdio.h>
02
03 int len_x = 8, total, x[] = {31, 57, 86, 75, 11, 52, 43, 17};
04
05 void sum()
06 {
07     for (int i=0; i<len_x; i++)
08         total += x[i];
09 }
10
11 int main()
12 {
13     total = 0;
14     sum();
15     printf("Total value is %d\n", total);
16     return 0;
17 }

```

3. (a) [10] Please complete the following program that calculate the number of negative integers and the number of odd integers: (You can write your answers on the answer sheet with the line number of the following program or you can put your answers directly on the problem sheet and make a note on the answer sheet)

```

01 #include <stdio.h>
02
03 int calculate(_____, int, _____);
04
05 int main()
06 {
07     int x[] = {3, -5, 8, 7, -11, 5, 4, -1}, odd, negative;
08     odd = calculate(x, sizeof(x)/_____, _____);
09     printf("# of odd numbers is %d\n", odd);
10     printf("# of negative numbers is %d\n", negative);
11     return 0;
12 }
13
14 int calculate(int data[], int len, _____negative)
15 {
16     int i, odd=0;
17     *negative = 0;
18     for (i=0; i<len; i++)
19     {
20         _____ += _____;
21         odd += _____;
22     }
23     return odd;
24 }

```

The output of the program execution is shown in the following figure:

```

# of odd numbers=6
# of negative numbers=3

```

- (b) [2] In C/C++ syntax, the type of a variable to save the value of an integer is *int*. What is the type of an integer pointer (the memory address of an integer variable) in C/C++? _____
- (c) [3] What variable does the program clear to zero in line 17 by the statement **negative=0*?
4. [10] In the following program, assume that there are *n* data items in the integer array defined by *int domino[40000][2]*; where $0 \leq n < 40000$. The *i*-th data item consists of two integers (*x*, *y*), representing two cards of a poker deck, where $0 \leq x, y < 52$, the data kept in *domino[i][0]* is *x* and the data kept in *domino[i][1]* is *y*. If two data items (*x*₁, *y*₁), (*x*₂, *y*₂) meet one of the conditions (*x*₁=*x*₂ and *y*₁=*y*₂) or (*x*₁=*y*₂ and *y*₁=*x*₂), then they are considered equivalent. Please complete the following program to calculate **the probability that two data items are equivalent when they are drawn randomly from the list**. In the next program, it calculates the amount of repetitions (*len*, line 23-27) of a pair out of $C(52,2)+52$ possible type of pairs. Then it calculates the possible number of pairs when it contains two equivalent data items and then accumulates to the variable *count* (line 28). The probability can be obtained by dividing this number by the number of random chosen pairs. (line 41) The function *numEquivDominoPairs* (line 10-31) expects to calculate the amount of pairs that contain equivalent data items. In order to efficiently use the library function *qsort* to pack those equivalent data items together in the array, the program normalizes each data item (line 13-19) by requiring *domino[i][0] ≤ domino[i][1]*. The program also prepares a comparison function *comp* (line 04-09) for *qsort* to compare the order for two items in which it compares the second integer in line 8 in case the first element is equal. Then the program uses a loop (line 23-27) to calculate the number of repetitions of equivalent items as *len*. **(You can write your answers on the answer sheet with the line number of the following program or you can put your answers directly on the problem sheet and make a note on the answer sheet)**

```

01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int comp(const void *a, const void *b)
05 {
06     int *arrayA = (int*) a, *arrayB = (int*) b;
07     int firstElemCompResult = arrayA[0] - arrayB[0];
08     return firstElemCompResult == 0 ? _____ : firstElemCompResult;
09 }
10 int numEquivDominoPairs(int dominoes[40000][2], int dominoesSize)
11 {
12     int i, tmp, count, len;
13     for (i=0; i<dominoesSize; i++)
14         if (dominoes[i][0]>dominoes[i][1])
15             {
16                 tmp = _____;
17                 dominoes[i][0] = _____;
18                 dominoes[i][1] = tmp;
19             }
20     qsort(_____, _____, _____, _____);
21     for (count=i=0; i<dominoesSize; i+=_____)
22     {
23         len = 1;
24         while (i+len<dominoesSize &&
25                 dominoes[i][0]==dominoes[_____][0] &&
26                 dominoes[i][1]==dominoes[_____][1])

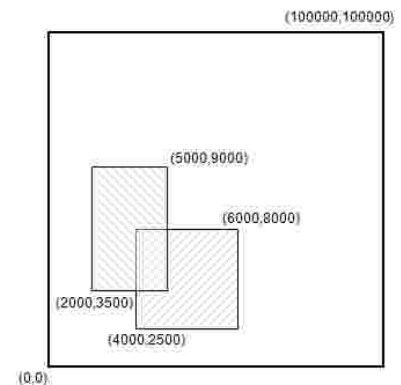
```

```

27         len++;
28         count += len * (len-1) / 2;
29     }
30     return count;
31 }
32
33 int main()
34 {
35     int i, n, dominoes[40000][2];
36
37     scanf("%d", &n);
38     for (i=0; i<n; i++)
39         scanf("%d%d", &dominoes[i][0], &dominoes[i][1]);
40
41     printf("%f\n", numEquivDominoPairs(dominoes,n)*2.0/n/(n-1));
42
43     return 0;
44 }

```

5. The figure on the right hand side shows two rectangles in the region (0,0) to (100000,100000) on the plane. The coordinate of the left-bottom corner of the rectangle is (x_1, y_1) and the coordinate of the right-top corner is (x_2, y_2) . A point (x, y) on the plane is inside the rectangle if both $x_1 \leq x < x_2$ and $y_1 \leq y < y_2$. Please write a program to calculate the area of the overlapped region inside both rectangles and the area inside only a single rectangle. For example, the coordinate of the left-bottom corner of the overlapped region is (4000,3500) and the coordinate of the right-top corner of the overlapped region is (5000,8000). According to the definition above, the area is $(4999-3999) * (7999-3499) = 4500000$. The area of the region inside only a single rectangle can be calculated by the areas of both rectangles. Please analyze the problem and write specified program segments following the directions:



- (a) [10] It seems easy to implement the program to calculate the areas for the above example. However, there could still be other geometric relations for two arbitrary rectangles in the region! If for the above example, we categorize them as the same when only one corner of a rectangle is inside the other rectangle and vice versa. Indeed, it could be the case that two rectangles do not overlap completely. Please plot the other five categories of geometries besides the one in the above figure.
- (b) [5] Please write a function `long long area(int rect[4])` that calculates and returns the area of a rectangle. The type `long long` is used because the area for a rectangle in the range of coordinates (0,0) to (100000,100000) could be too large for an `int` type of variable to store. Please use a 4-element integer array `rect` to store the four coordinate values x_1, y_1, x_2, y_2
- (c) [5] As specified by the example shown in the figure. When the program reads four sets of coordinates $(x_1, y_1)-(x_2, y_2)$, $(x_3, y_3)-(x_4, y_4)$ of the two rectangle to the array variable `int rect[2][4]`; (the variable `rect[0][0]`, `rect[0][1]`, `rect[0][2]`, `rect[0][3]` stores x_1, y_1, x_2, y_2 , respectively and the variable `rect[1][0]`, `rect[1][1]`, `rect[1][2]`, `rect[1][3]` stores x_3, y_3, x_4, y_4 respectively). The main goal of the program is to calculate the width as $\min(x_2, x_4) - \max(x_1, x_3)$ and height $\min(y_2, y_4) -$

$\max(y_1, y_3)$ for the overlapped region and calculate the area accordingly. This area is saved to variable **overlappedArea**. The area of the region inside only a single rectangle can be calculated by the areas of both rectangles and save to the variable **nonOverlappedArea**. Please write two functions `int min(int x, int y)` and `int max(int x, int y)` to calculate the smaller one and the larger one out of any two numbers x, y . Using these two functions to write a program that calculate the width and height of the overlapped region and then a segment of program to calculate `overlappedArea` and `nonOverlappedArea`

(d) [5] There could be a couple of ways to continue the design. For example one could calculate the number of corners of one rectangle to be inside another rectangle such that the program could determine which category of part (a) the input data belongs. Then calculate the area for the overlapped region. Therefore, one need to write a function `int ptInRect(int point[2], int rect[4])` to determine if a point is inside a rectangle and write a function `int cornersInRect(int rect1[4], int rect2[4])` to determine the number of corners of a rectangle inside the other rectangle, which will returns with one result out of the set $\{0, 1, 2, 4\}$. In the main function, the above functions can be used to calculate `count1 = cornersInRect(rect1, rect2);` and `count2 = cornersInRect(rect2, rect1);`. Then use these two variables `count1` and `count2` to determine the category of geometry. Although this method is direct and should be able to give the correct answer, the amount of codes is big for this midterm test. In the following, a more concise method is explained to calculate the area for the overlapped region. Please write a function `int isOverlapped(int rect1[4], int rect2[4])` to determine if two rectangle has any overlapped region. (Hint: the rule is like the right side of the first rectangle is smaller than the left side of the second rectangle, ... etc) If two rectangles do not have any overlapped region, the area `overlappedArea` would be zero.

(e) [5] Following is a very crucial observation: let the two rectangles be specified by the four set of coordinates (x_1, y_1) , (x_2, y_2) , (x_3, y_3) , (x_4, y_4) . No matter which categories of part (a) the two rectangles belong, the larger one of the left side boundaries of two rectangles x_1, x_3 is the left side boundary of the overlapped region and the smaller one of the right side boundaries of two rectangles x_2, x_4 is the right side boundary of the overlapped region. Therefore the width of the overlapped region is still $\min(x_2, x_4) - \max(x_1, x_3)$. Similarly, the larger one of the bottom side boundaries of two rectangles y_1, y_3 is the bottom side boundary of the overlapped region and the smaller one of the top side boundaries of two rectangles y_2, y_4 is the top side boundary of the overlapped region. Therefore the height of the overlapped region is still $\min(y_2, y_4) - \max(y_1, y_3)$. Please use the functions completed in part (c) and (d) to write a segment of codes to calculate `overlappedArea` and `nonOverlappedArea`.

6. Please point out and correct the syntax or semantic errors (caused by syntax errors or syntax trap) in the following program segments

(a) [6] The following program segment wishes to print out messages to a student in two cases when he/she gets a score not less than 80 or less that 60. However, the program does not behave as he/she expected originally. What is the output when input a score 75? What is the output when input a score 50? Please make minimal modifications to the code to achieve the expected results.

```

01  int score;
02  scanf("%d", &score);
03  if (score>=60)
04      if (score>=80)
05          printf("Congratulation, you get an A.\n");
06  else
07      printf("I am sorry that you failed.\n");

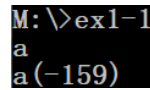
```

- (b) [3] The following program segment expects to input an arbitrary character and print out the corresponding ASCII encoding:

```

01 int ch=-1;
02 scanf("%c", &ch);
03 printf("%c(%d)", ch, ch);

```



```

M: \>ex1-1
a
a(-159)

```

But the program outputs -159 instead of 97 as the figure shows. Why is this happening? How do we correct this?

- (c) [3] The following program segment expects to find out the maximum of 10 input numbers. However, it does not always output the maximum of the input. What is the problem? How do we correct it?

```

01 int i, max, data;
02 for (i=0; i<10; i++)
03 {
04     scanf("%d", &data);
05     if (data>max)
06         max = data;
07 }
08 printf("Maximum is %d\n", max);

```

- (d) [3] What do you expect the following code segment to output? Will there be any peculiarity in executing this segment? How do you correct this?

```

01 double x=0.3;
02 for (x=0.3; x != 1.0; x+=0.1)
03     printf("%f\n", x);

```