1131 NTOUCSE 程式設計 1C 期中考

姓名：_____ 系級：_____ 學號：_____ 113/10/29 (二)

考試時間：**13:20 – 16:00**

Exam rules：
1. **Close book**, **close** everything including quizs, 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**.
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. Please answer the following programming questions:

(a) [5] Please write a segment of codes, read in the data shown in the following figure, where the integer is decimal 10~15 digits, add them and print in the field of 20 characters aligning rightward. In the following figure, □ represents a space. Please use suitable datatype to define two variables x and y, use scanf() to read in data, and use printf() to output the sum of two numbers.

```
123456789012 □□+□□87654321
□□□□□□□□123544443333
```

(b) [5] The following code segment uses printf() to output the data in the format shown below in the figure. The first four characters represent a number in hexadecimal format. You have to fill in exactly 4 characters with 0, i.e. no leading blanks, following by a comma, then a floating number with 6 integers and 3 fraction digits. Please use suitable format conversion commands for printf()

```
003f,□□□□1.234
```

```
01 int x;
02 double y;
03 scanf("%d%lf", &x, &y); // read decimal integer in the range 0~65535 and real number in the range 0～999999
04 printf("_____", x, y);
```

(c) [10] Consider the data in the following figure. The loop statement in the following wants to read in a decimal digit for each repetition and add them together. This sum is printed out at line 06. How do you do this with scanf() in line 03? How about doing the same with getchar()?

```
1234
```

```
01 int i, x, sum;
02 for (sum=i=0; i<4; i++) {
03         _____
04         sum += _____
05 }
06 printf("sum is %d\n", sum); // output 10 for the above input data
```

(d) [5] The following code segment wishes to read one character a time from the input stream. It can be compiled correctly and can be executed. However, it always ends prematurely before all the data are exhausted. What happened? How do you fix it?

```
01 char c;
02 while ((c=getchar())!=EOF) {
03         …
```

```
04 }
```

(e) [5] The following code segment wants to read the two lines of data shown in the following figure. The first row is an integer, the character at the second row is having trouble to be read in correctly. What happens with this code? How do we use scanf() correctly to solve this problem?

```
01 int x;
02 char ch;
03 scanf("%d", &x);
04 scanf("%c", &ch);
```

```
1234
y
```

(f) [5] The following code segment always caused some illegal memory access while executed. Why does this happen? This code can easily result to a vulnerability for the system such that hackers can gain the control of the system. If one insists to use scanf() in reading string in the data stream, what is the simplest method to avoid this situation?

```
01 char buf[10];
02 scanf("%s", buf);
```

2. Please finish the following code snippet according to the requirement in the problem statement.

(a) [6] Please use integer division and remainder to finish the following the code segment to output the 32 bits of a decimal integer reversely on the screen (the least significant bit first).

```
01 int x=13;
02 printf("decimal: %d => binary: ", x);
03 for (; x>0; _____)
04         printf("%d", _____);
05 printf("\n");
```

Program output:
decimal: 13 => binary: 1011

(b) [6] Following (a), use an additional array to keep all the bits such that the most significant bit can be printed first.

```
01 int i, b[32], x=13;
02 for (i=0; x>0; _____)
03         _____ = _____;
04 while (i-->0)
05         printf("%d", b[i]);
06 printf("\n");
```

Program output:
decimal: 13 => binary: 1101

(c) [6] Following (b), write a recursive function to print the most significant bit first without using an additional array as in part (b).

```
01 void printBinary(const int x) {
02         _____
03         printBinary(_____);
04         _____
05 }
06 int x=13;
07 printBinary(x);
08 printf("\n");
```

Program output:
decimal: 13 => binary: 1101

(d) [6] Following part (b) and part (c), how can one print the most significant bit first but does not want to use an array nor a recursive function. The following code segment raises 2 to k-th power as w such that it satisfies 2*w>x>=w. Then this w is used to calculate the most significant bit of x before it is divided by 2 each time in a loop and controls the execution of the loop.

```
01 int w, x2, x=13;
02 printf("decimal %d ==> binary ", x);
03 for (w=1,x2=x; x2_______1; x2/=2)
04         w*= _________;
05 for (; w>0; x%=w,w/=2)
06         printf("%d", _____________);
07 printf("\n");
```

Program output:
decimal: 13 => binary: 1101

3. [10] There is an integer array storing ascending integers like 1,1,1,3,4,4,5,7,11,11,15. Please finish the following code segment to eliminate all duplicated items in the array. For the above example, the remaining data in the array are 1,3,4,5,7,11,15. The outputs of line 10 and 11 are as follows:

```
1 3 4 5 7 11 15
```

```
01 #include <stdio.h>
02 int main() {
03         int i,j;
04         int x[]={1,1,1,3,4,4,5,7,11,11,15};              // use x[i] to check each element in the array
05         for (j=0,i=1; i<sizeof(x)/_____________; i++)    // use x[j] to store data after duplication are removed
06                 if (_________________)                   // note x[j] is either less than or equal to x[i]
07                         x[_______] = x[i];
08         for (i=0; ____________; i++)
09                 printf("%d%c", x[i],_________________); // except the new line character at the end of all data
10         return 0;                                        // a space is printed between any two numbers
11 }
```

4. [20] The sequence of a DNA segment consists of the combination of the first letter of the chemical bases {A, T, C, G}. Now we want to print a large number of DNA sequences according to the inversion number of each DNA sequence. We want to write a function that can calculate the inversion number of a DNA sequence, for example the inversion number of sequence “ATCGA” is 5, which is the number of inverted pairs TC, TG, TA, CA, GA. The order of an English letter is ABCDE…XYZ. An inverted pair, e.g. TC, is a pair of letters such that their order is different from the above natural order or letters. In general, the inversion number of a length-n sequence can be calculated by two layers of loops to enumerate all C(n,2) combinations and compare their ASCII values. However, this is time consuming for large amount of DNA sequences while a DNA sequence is so special that it contains only four kind of letters. the following program uses an easier way to calculate the inversion number for a DNA sequence that processes the DNA sequence backwards, starting from the last letter: use three integer variables ***a***, ***ac***, and ***acg*** to recording the amount of A, C, G letters seen up to the current processed letter (i.e. following the current processed letter), in which the variable ***a*** records the amount of letter A, the variable ***ac*** records the amount of both letters A and C, the variable ***acg*** records the amount of all three letters A, C, and G. When you see a letter T in the DNA sequence, the inversion number is the value kept by the variable ***acg***. When you see a letter G in the DNA sequence, the inversion number is the value kept by the variable ***ac***. When you see a letter C in the DNA sequence, the inversion number is the value kept by the variable ***a***. Please finish the following

function to calculate the inversion number of a DNA sequence:

```
01 #include <stdio.h>
02 int inversionNumber(char DNA[]) {
03        int inv, a, ac, acg, n, i;
04        n=0; while (_____!=0) n++;            // calculate number of characters in the DNA[] array
05        for (_____,i=n-1; i>=0; i--) { // initialize variables inv, a, ac, acg
06              switch (_____) {
07                    case 'T':
08                          inv += _____;
09                          break;
10                    case 'G':
11                          inv += _____;
12                          _____++;
13                          break;
14                    case 'C':
15                          inv += _____;
16                          _____++, _____++;
17                          break;
18                    case 'A':
19                          _____++, _____++, _____++;
20              }
21        }
22        return _____;
23 }
24 int main() {
25        char DNA[] = "ATCGA";
26        printf("inversion number=%d\n", inversionNumber(DNA));
27        return 0;
28 }
```

5. [11] Please finish the following loop to merge two lists sorted ascendingly in the arrays data1[], data2[] to a new array data[]. The resulting data remains in the ascending order.

```
01 #include <stdio.h>
02 int main() {
03        int data1[]={6, 9, 11, 12, 15, 18};
04        int data2[]={1, 2, 3, 7, 14, 16, 20};
05        int data[100];
06        int i, idx1, idx2, idx, len1=6, len2=7, len;
07        idx = idx1 = idx2 = 0;
08        len = len1+len2;
09        while (_____ < len){
10              if (_____ < len1 && _____ < len2){
11                    if (data1[idx1] < data2[idx2])
12                          _____ = _____;
13                    else
14                          _____ = _____;
15              }
16              else if (idx1 < len1)
17                    _____ = _____;
18              else
19                    _____ = _____;
20        }
21        printf("len=%d: ", len);
22        for (i=0; i<idx; i++)
23              printf("%d%c", data[i], " \n"[i==idx-1]);
24        return 0;
25 }
```