國立中正大學
109 學年度碩士班招生考試
試 題

[第 2 節]

科目名稱	計算機概論(含程式設計)
系所組別	資訊工程學系-乙組

1. (12%) Assume that the following program is executed on a little-endian machine. Write down the output of the following C program.

```c
#include <stdio.h>
#define LOC(row, col) row*3+col

int main(void)
{
    int A[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
    int *ptr1;
    char *ptr2;

    ptr1 = (int*) A;
    printf("%d\n", *ptr1);
    printf("%d\n", *(ptr1+LOC(1,1)));
    printf("%d\n", ptr1[LOC(2,1)]);
    printf("%d\n", ptr1[5]/5);

    ptr2 = (char*) A;
    printf("%d\n", ptr2[0]);
    printf("%d\n", ptr2[1]);

    return 0;
}
```

2. (8%) Write down the output of the following C program.

```c
#include <stdio.h>

void func(int **a, int **b)
{
    int *temp;

    temp = *a;
    *a = *b;
    *b = temp;
}

int main(void)
{
    int **x1, *y1, z1, **x2, *y2, z2;

    z1 = 10, z2 = 20;
    x1 = &y1;
    y1 = &z1;
    x2 = &y2;
    y2 = &z2;
    printf("%d %d %d %d\n", *y1, *y2, **x1, **x2);
    func(x1,x2) ;
    printf("%d %d %d %d\n", *y1, *y2, **x1, **x2);
    return 0;
}
```

3. (30%) A memory leak is a dynamically allocated-memory object that is not deallocated properly and cannot be accessed again. A double-free error happens when a deallocated-memory object is freed again. These issues often occur in programming languages like C and C++.

(a) (21%) There are three code fragments in (1), (2), and (3). L1~L17 are line numbers. Do the code fragments (1), (2), and (3) exist memory-leak or double-free problems at run-time? If yes, please move the existed free() statement or insert a free() statement to correct the problems. The modified programs do not have any memory-leak or double-free errors.

(Answer sample: insert a statement "free (...) " immediately after L1)

(b) (9%) Java programming language provides a garbage collection technique to avoid memory-leak errors. What is garbage collection?

```
L1  #include <stdio.h>              (1)
L2  #include <stdlib.h>
L3
L4  int main(void)
L5  {
L6    int *ptr;
L7
L8    ptr = (int*) malloc(10);
L9    *ptr = 2020;
L10   printf("%d\n", *ptr);
L11   ptr = (int*) malloc(20);
L12   ptr[0] = 10;
L13   ptr[1] = 20;
L14   printf("%d %d\n", *ptr, ptr[1]);
L15   free(ptr);
L16   return 0;
L17 }
```

```
L1  #include <stdio.h>              (2)
L2  #include <stdlib.h>
L3  int main(void)
L4  {
L5    int *ptr;
L6    int i;
L7
L8    for (i=0; i<100; i++) {
L9      ptr = (int*) malloc(10);
L10     *ptr = i+1;
L11     printf("%d\n", *ptr);
L12   }
L13   free(ptr);
L14
L15   return 0;
L16 }
L17
```

```
L1  #include <stdlib.h>             (3)
L2  int main(void)
L3  {
L4    int *ptr;
L5    int i;
L6
L7    for (i=0; i<100; i++) {
L8      ptr = (int*) malloc(10);
L9      *ptr = i+1;
L10     if (i%2 == 0) {
L11       *ptr ++;
L12       free(ptr);
L13     }
L14   }
L15   return 0;
L16 }
L17
```

4. (10%) Please answer the following questions for the two binary trees in the following figure.

(a) (4%) For each tree in the following figure, list the leaf nodes and the non-leaf nodes.

(b) (6%) Write out the inorder, preorder, and postorder traversals for each tree in the following figure.
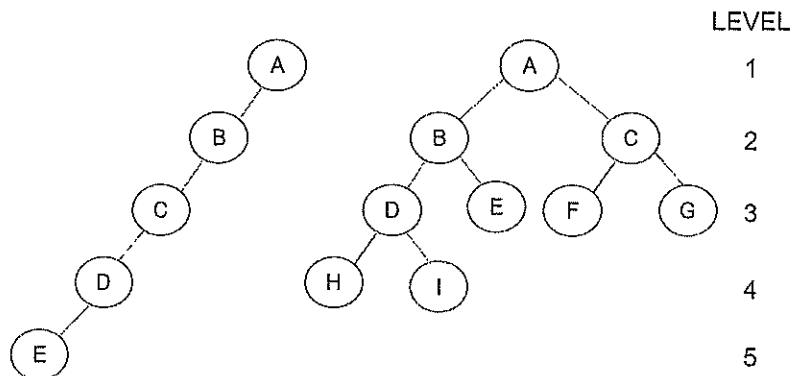
5. (20%) Carefully examine the following twenty sentences and give your answer, true (T) or false (F), for each sentence. Two points are given for each correct answer, whereas **three points are deducted for each incorrect answer**. Therefore, please do not guess at the answer if you are unsure since **the expected value is negative**. Note that the deduction for incorrect responses only affects the 20 points of this question.

   (a) The fields of a user-defined data type "union" in C share their memory space.

   (b) The maximum number of nodes in a binary tree of depth 10 is 1024.

   (c) We can invert a linked list in place (i.e., no need to use additional memory with a size proportional to the elements in the linked list).

   (d) A heap (i.e., priority queue) is better than a binary search tree when we examine whether an element is stored in the data set.

   (e) Dijkstra algorithm cannot deal with the graph with negative edge weights.

   (f) In most cases, prime numbers are good divisors for implementing a division-based hash function.

   (g) A balanced binary search tree is always the best binary search tree for storing the records even if the identifiers are with different searching probabilities.

   (h) Quick sort is faster than insertion sort even if the input size is small.

   (i) A stack is a first-in-first-out data structure.

   (j) A user-defined "struct" in C creates a data type that can be used to group items of possibly different types into a single type.

6. (20%) Consider the following graph. Please write down the sequences of edges selected by the following two algorithms, respectively.

   (a) (10%) Kruskal's algorithm.

   (b) (10%) Prim's algorithm (start from vertex a).