國立臺南大學 109 學年度 **資訊工程學系碩士班** 招生考試　計算機概論　試題卷

1. Design arbitrary classes, including at least one parent class and one child class, in **Object-Oriented programming language** (JAVA/C++) to demonstrate,
   A. Data encapsulation,
   B. Overloading,
   C. Overwriting,
   D. Polymorphism, and
   E. Late binding.　(15%)

2. Write down the following requirements in *C/ C++* programming language.
   A. Create the following **irregular 2D array**,　(15%)

83	2	64	23	15
62	37			
59	45	78	90	
83				

   Using nested ***for*** loop to compute and print,
   (1) The Irregular 2D array
   (2) The number of elements in the 2D array
   (3) The average of all elements
   (4) The Maximum and its index
   (5) The Minimum and its index

   B. Implement *m* x *n* as the function *int multiply(int m, int n)* by **recursion**.　(10%)

   C. Implement the **recursive function** *int count(char ch, const char* str)* to count the occurrence of a character *ch* in a string *str*. (i.e., the times of *ch* appeared in *str*.)　(10%)

3. Using the KMP pattern matching algorithm. Please find the failure values for the pattern p[ ]=ababbababaa. (10%)

4. Please use a method with complexity O(1) to collect a used circular linked list represented polynomials pointed by ***ptr***. The existing available list space is pointed by ***avail***. Please fill instructions in the **block** of **Fig.1**.　(10%)

```
void cerase(polyPointer *ptr)
{
    polyPointer temp;
    if
    }
}
```

Fig. 1

5. Applying Set Representation and Weighting Rule to solve the Equivalence Classes Problem for the following equivalence pairs:
$0 \equiv 4, 3 \equiv 1, 6 \equiv 10, 8 \equiv 9, 7 \equiv 4, 6 \equiv 8, 3 \equiv 5, 2 \equiv 11, 11 \equiv 0$. (註：當二個集合的成員個數一樣時，以樹根編號較小者為合併後的樹根) (10%)

6. Please fill 4 instructions into the **block** of **Fig. 2** to invert a single linked list. (10%)

```
listPointer invert(listPointer lead)
{
    listPointer middle, trail;
    middle = NULL;
    while (lead) {
        [          ]
        [          ]
        [          ]
    }
    return middle;
}
```

Fig. 2

7. Please apply the Floyd-Warshall Algorithm to Fig.3 and write out the corresponding cost adjacency matrices A-1, A0, A1, and A2. (10%)

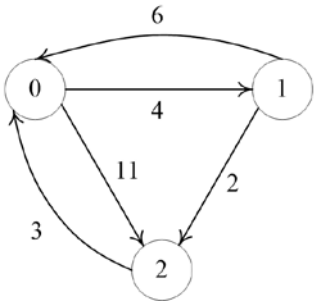

Fig. 3