國立中興大學

110 學年度

碩士班考試入學招生

試　　題

學系：資訊科學與工程學系

　　　乙組

科目名稱：資訊系統

Part I (50%)

1. (Single Choice) (4 pts) Which of the following operations is not O(1) for an array of data? Assume that the array elements have been sorted and are all distinct.
   (a). Find the i-th largest element
   (b). Delete a number, *x*, in the array
   (c). Find the i-th smallest element
   (d). All of the above
2. (Single Choice) (4 pts) Consider the following pseudocode. What is output for input "geeksquiz"?

```
Declare a stack of characters
while ( there are more characters in the word to read )
{
    read a character
    push the character on the stack
}
while ( the stack is not empty )
{
    pop a character off the stack
    write the character to the screen
}
```

   (a). geeksquizgeeksquiz
   (b). ziuqskeeg
   (c). geeksquiz
   (d). ziuqskeegziuqskeeg
3. (Single Choice) (4 pts) Following is C like pseudo code of a function, called fun(), that takes a number, i.e., *n*, as an argument, and uses a stack *S* to do processing.

```
void fun(int n)
{
    Stack S;    // Create an empty stack S
    while (n > 0)
    {
        // Pushes the value of n%2 to stack S, %: modulo (remainder) operator
        push(&S, n%2);
        n = n/2;
    }

    while (!isEmpty(&S))          // Run while Stack S is not empty
        printf("%d ", pop(&S));   // pop an element from S and print it
}
```

What does the above function do in general?
(a). Prints binary representation of n in reverse order.
(b). Prints binary representation of n
(c). Prints the value of Logn
(d). Prints the value of Logn in reverse order

4. (Single Choice) (4 pts) Following is C like pseudo code of a function that takes a Queue as an argument, and uses a stack S to do processing.

```
void fun(Queue *Q)
{
  Stack S; // Create an empty stack S

  // Run while Q is not empty
  while (!isEmpty(Q))
  {
    // deQueue an item from Q and push the dequeued item to S
    push(&S, deQueue(Q));
  }

  // Run while Stack S is not empty
  while (!isEmpty(&S))
  {
     // Pop an item from S and enqueue the poppped item to Q
     enQueue(Q, pop(&S));
  }
}
```

What does the above function do in general?
(a). Removes the last from Q
(b). Keeps the Q same as it was before the call
(c). Makes Q empty
(d). Reverses the Q

5. (Single Choice) (4 pts) Consider the following function, func(), that takes reference to head of a Doubly Linked List as parameter. Assume that each node of the doubly linked list has previous pointer as prev and next pointer as next.

```
void fun(struct node **head_ref)
{
  struct node *temp = NULL;
  struct node *current = *head_ref;

  while (current != NULL)
  {
    temp = current->prev;
    current->prev = current->next;
    current->next = temp;
    current = current->prev;
  }

  if(temp != NULL )
     *head_ref = temp->prev;
}
```

Assume that reference of head of following doubly linked list is passed to above function 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the modified linked list after the function call?

(a). 2 <--> 1 <--> 4 <--> 3 <--> 6 <-->5
(b). 6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1.
(c). 5 <--> 4 <--> 3 <--> 2 <--> 1 <-->6.
(d). 6 <--> 5 <--> 4 <--> 3 <--> 1 <--> 2

6. (5 pts) Design an algorithm that takes two arrays, array1 and array2, and returns true if the arrays are disjoint, i.e. have no elements in common. You may freely use standard data structures and algorithms in your solution, without explaining how they are implemented. Write down your algorithm as pseudocode and your algorithm should take O(nlogn) time.

7. (4 pts) Please fill the blanks (1) and (2) in English.

How does the controller inform OS that it has finished its operation? The complete protocol for interaction between the host and a controller can be intricate, but the basic handshaking notion is simple. We explain handshaking with an example. The controller indicates its state through the busy bit in the status register. The controller sets the busy bit when it is busy working and clears the busy bit when it is ready to accept the next command. The host signals its wishes via the command-ready bit in the command register. The host sets the command-ready bit when a command is available for the controller to execute. For this example, the host coordinates with the controller by handshaking as follows.

(1) The host repeatedly reads the busy bit until that bit becomes clear.
(2) The host sets the write bit in the command register and writes a byte into the data-out register.
(3) The host sets the command-ready bit.
(4) When the controller notices that the command-ready bit is set, it sets the busy bit.
(5) The controller reads the command register and sees the write command. It reads the data-out register to get the byte and does the I/O to the device.
(6) The controller clears the command-ready bit, clears the error bit in the status register to indicate that the device I/O succeeded, and clears the busy bit to indicate that it is finished.

This loop is repeated for each byte. The technique in Step (1) is called (1): it is in a loop, reading the status register over and over until the busy bit becomes clear. If the controller and device are fast, this method is a reasonable one. But if the wait may be long, the host should probably switch to another task. How, then, does the host know when the controller has become idle? For some devices, the host must service the device quickly, or data will be lost. For instance, when data are streaming in on a serial port or from a keyboard, the small buffer on the controller will overflow and data will be lost if the host waits too long before returning to read the bytes. But (1) becomes inefficient when it is attempted repeatedly yet rarely finds a device ready for service, while other useful CPU processing remains undone. In such instances, it may be more efficient to arrange for the hardware controller to notify the CPU when the device becomes ready for service, rather than to require the CPU to poll repeatedly for an I/O completion. The hardware mechanism that enables a device to notify the CPU is called an (2).

8. (6 pts) Figure 1 show the process state transition diagram. Please fill the blanks of (1), (2), and (3).

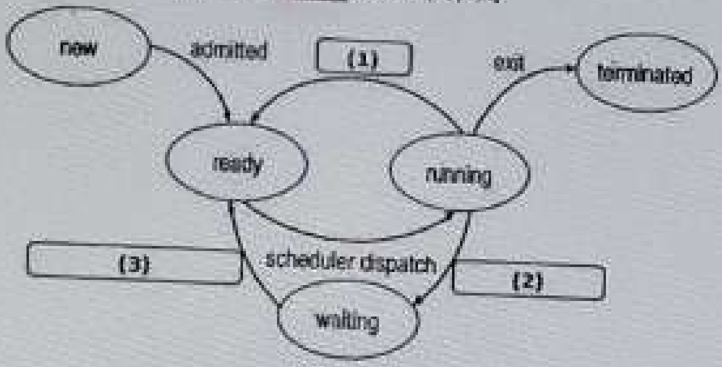

Figure 1.

9. (6 pts) If virtual address space is 8 pages with each page size is 4 KB. As shown in Fig. 2, assume a process's size is only 4 pages. If the process issues a memory access to page 6 with offset 0. Obviously, this access is illegal. (a) How to detect this illegal access? (b) And by whom?

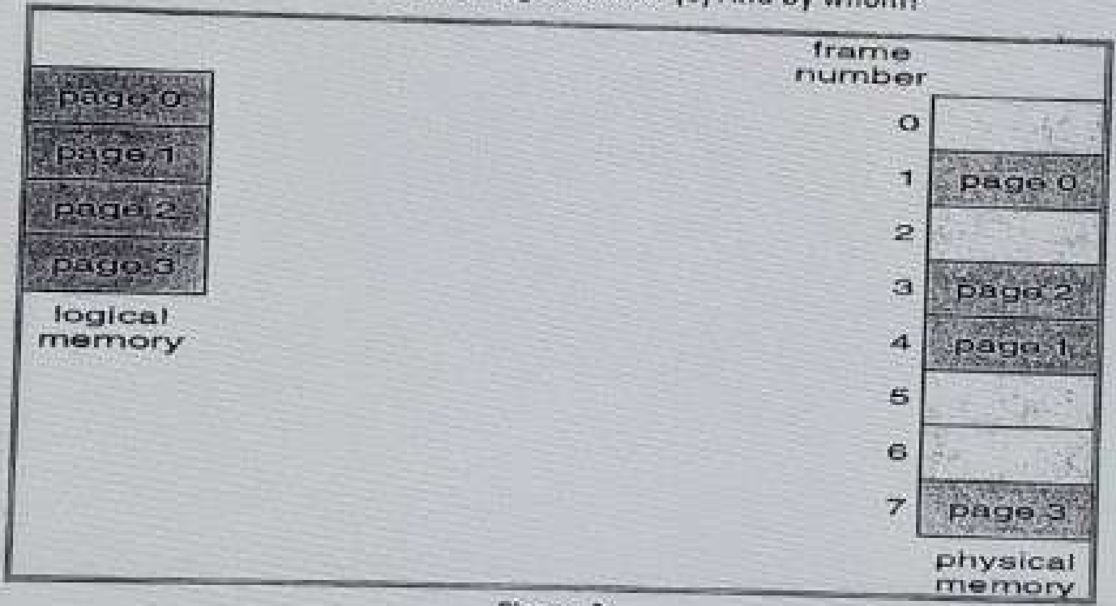

Figure 2.

10. (5 pts) A classical solution to race condition is to associate a variable called *lock*. Then, we can use the following code for checking if we can enter critical section or not. However, this solution is not correct and can cause race condition. Why?

```
lock = 0;
while (lock != 0) ;
lock = 1;
      Critical Section
lock = 0;
```

Figure 3.

11. (4 pts) User processes access disks via file abstract, i.e., by the pair of (filename, offset). However, disks are in fact consists of a number of blocks (or sectors) and are accessed by logical block number (or logical block address). Thus, the file system must transfer from the (file, offset) to the corresponding logical block number. That is, given a (filename, offset), file system must find the

corresponding location, i.e., logical block address, in the disks. Please briefly describe how the file system perform the transformation.

PART II (50%)

12. Confusion and Diffusion are two properties of the operation of a secure cipher. Please explain the difference between Confusion and Diffusion. (3%)

13. Data Encryption Standard (DES) is a Feistel cipher. Please draw the structure of Feistel cipher (8%) and write down how many rounds does the DES perform (2%).

14. Please write down four operations of each round in the Advanced Encryption Standard (AES) algorithm. (4%)

15. The AES algorithm uses Galois field theory. In Galois field GF(2^8), the multiplication (denoted by •) corresponds with the multiplication of polynomials modulo an irreducible polynomial of degree 8. For the AES algorithm, this irreducible polynomial is $x^8+x^4+x^3+x+1$.
    a. Please calculate the result of {67}•{CB}. (4%)
    b. Please calculate the result of {57}•{83}. (4%)

16. Please write down the best, average, and worst case time complexity of the following sorting algorithms.
    a. Insertion sort. (3%)
    b. Quick sort. (3%)
    c. Merge sort. (3%)
    d. Heap sort. (3%)

17. Please explain stable sorting and in-place sorting. (4%)

18. True or false:
    a. The Dijkstra's algorithm (single run) solves the single-source shortest path problem with positive, zero, negative edge weights, if there is no negative cycle. (1%)
    b. The Bellman-Ford algorithm (single run) solves the all-pairs shortest path with positive, zero, or negative edge weights, even if there are negative cycles. (1%)
    c. The Floyd-Warshall algorithm (single run) solves the all-pairs shortest path problem with non-negative edge weights. (1%)

19. Please explain the difference between P problems and NP problems. (6%)