1. The order of the nodes visited in a binary tree T using preorder traversal is G, A, H, I, B, L, J, E, D, F, K, C, M. If inorder traversal is used, it is H, I, A, L, B, J, G, D, F, E, K, M, C. Please draw the binary tree T. (20%)

2. If S is $O(1)$, what is the worst case order of each of the following program segments. (20%)
   (a) for (k = 0; k < n; k++)
   　　　for (j = 0; j < k; j++)
   　　　　S;
   (b) for (i = 0; i < n; i++)
   　　　for (j = n; j > i; j--)
   　　　　for (k = i; k < j; k++)
   　　　　　S;

3. (1) A queue may be implemented by a linked list or by an array. Please compare these two situations and explain their advantages and disadvantages.
   (2) A stack may be implemented by a linked list or by an array. Please compare these two situations and explain their advantages and disadvantages. (20%)

4. Suppose you have an array of n integers. Each of them is either 0 or 1. Give an $O(n)$ algorithm to rearrange the list so that all 0s precede all 1s. For example, the list 0100110001 should be transformed into 0000001111. You may use only constant extra space. (15%)

5. Suppose you have an array of n integers. Each of them is 0, 1 or 2. Give an $O(n)$ algorithm to rearrange the list so that all 0s precede all 1s, which in turn precede all 2s. You may use only constant extra space. (10%)

6. With the following declaration, the function insert_max_heap performs an insertion into a max heap that contains n elements. Please complete the function by filling out the three blanks. (15%)

```
#define MAX-ELEMENTS 200 /* maximum heap size+1 */
#define HEAP-FULL(n) (n == MAX-ELEMENTS-1)
#define HEAP-EMPTY(n) (!n)
typedef struct {
        int key;
        /* other fields */
        } element;
element heap[MAX-ELEMENTS];
int n = 0;
```

```
void insert-max-heap(element item, int *n)
{
/*insert item into a max heap of current size *n */
    int i;
    if (HEAP-FULL(*n)){
        fprintf(stderr, "The heap is full. \n");
        exit(1);
    }
    i = ++(*n);
    while (          (1)          ) {
          (2)
          (3)
    }
    heap[i] = item;
}
```