※ 考生請注意：本試題不可使用計算機。　請於答案卷(卡)作答，於本試題紙上作答者，不予計分。

1. The followings are the array transpose and fast_transpose functions, please complete the parts indicated by XXX. (15%)

# of rows (columns)

# of nonzero terms

	row	col	value
a[0]	6	6	8
[1]	0	0	15
[2]	0	3	22
[3]	0	5	-15
[4]	1	1	11
[5]	1	2	3
[6]	2	3	-6
[7]	4	0	91
[8]	5	2	28

transpose →

	row	col	value
b[0]	6	6	8
[1]	0	0	15
[2]	0	4	91
[3]	1	1	11
[4]	2	1	3
[5]	2	5	28
[6]	3	0	22
[7]	3	2	-6
[8]	5	0	-15

```
void transpose (term a[], term b[])
/* b is set to the transpose of a */
{
    int n, i, j, currentb;
    n = a[0].value;   /* total number of elements */
    b[0].row = a[0].col;   /* rows in b = columns in a */
    b[0].col = a[0].row;   /*columns in b = rows in a */
    b[0].value = n;
    if (n > 0) {                    /*non zero matrix */
        currentb = 1;
        for (i = 0; i < a[0].col; i++)
        /* transpose by columns in a */
              for( j = 1; j <=   n; j++)
              /*   find elements from the current column */
              if (a[j].col == i) {
             /* element is in current column, add it to b */
                 XXX-1;
                 XXX-2;
                 XXX-3;
                 currentb++
              }
    }
}
```

2. Complete the parts indicated by XXX for stack and queue operations (10%)

```
void add(int *top, element item)
{
 /* add an item to the global stack */
      if (*top >= MAX_STACK_SIZE-1)   {
            stack_full( );
            return;
      }
      stack[XXX-1] = item;
}

element delete(int *top)
{
 /* return the top element from the stack */
      if (*top == -1)
          return stack_empty( );   /* returns and error key */
      return stack[(XXX-2);
 ]

void addq(int *rear, element item)
{
/* add an item to the queue */
     if (*rear == MAX_QUEUE_SIZE_1) {
         queue_full( );
         return;
     }
   queue [XXX-3] = item;
}
```

3. Translate the expression $(a+b)*c-d/(e+f/g)+h$ into postfix form by stack approach. You must show the stack status step by step (10%)

4. The following is the *attach* function for attaching an item to a polynomial. Please complete the parts indicated by XXX. (10%)

```
typedef struct poly_node *poly_pointer;
typedef struct poly_node {
      int coef;
      int expon;
      poly_pointer link;
};

void attach(int coefficient, int exponent, poly_pointer *ptr)
{
/* create a new node attaching to the node pointed to by ptr. ptr is updated to point to this new node. */
      poly_pointer temp;
      temp = (poly_pointer) malloc(sizeof(poly_node));
      if (IS_FULL(temp)) {
            fprintf(stderr, “The memory is full\n”);
            exit(1);
      }
      temp->coef = XXX-1;
      temp->expon = XXX-2;
      XXX-3 = temp;
      XXX-4= temp;
}
```

5. Complete the parts indicated by XXX of the function (concatenating singly linked lists, list ptr1 followed by the list ptr2). (10%)

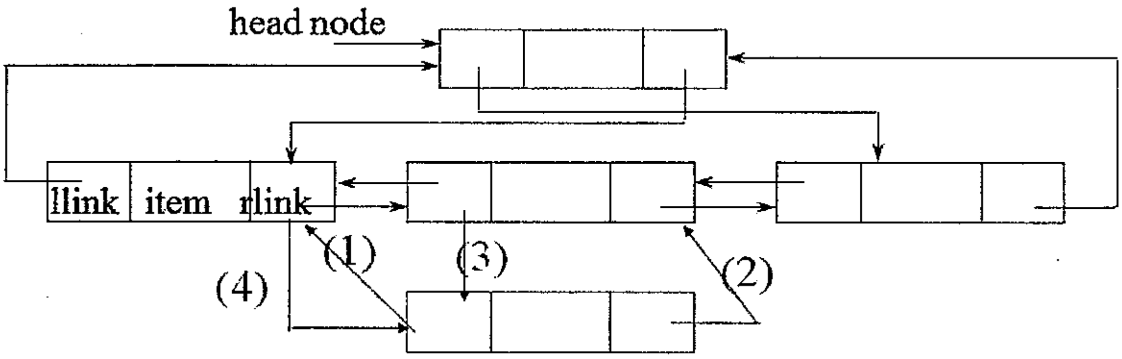
```
typedef struct list_node *list_pointer;
typedef struct list_node {
      char data;
      list_pointer link;
};

list_pointer concatenate(list_pointer ptr1, list_pointer ptr2)
{
   list_pointer temp;
   if (IS_EMPTY(ptr1)) return ptr2;
   else {
    if (!IS_EMPTY(ptr2)) {
      for (temp=XXX-1;XXX-2;temp=temp->link);
      temp->link = XXX-3;
    }
    return ptr1;
  }
}
```

6. Complete the function of inserting a newnode into a doubly linked circular list. (10%)

```
typedef struct node *node_pointer;
typedef struct node {
    node_pointer llink;
    element item;
    node_pointer rlink;
}

void dinsert(node_pointer node, node_pointer newnode)
{
    (1) ;
    (2) ;
    (3) ;
    (4) ;
```

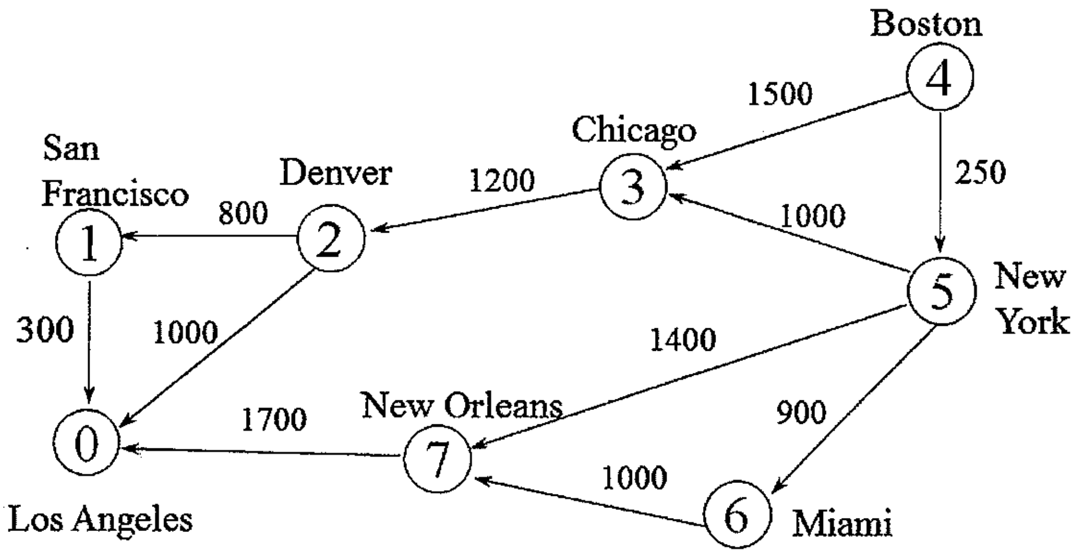


```
}
```

7. For the sequence: 27, 6, 38, 2, 60, 12, 58, 16, 47, 18 ; Please write down every sequence of each step while applying MAX Heap Sort, Quick Sort, Interactive Merge Sort, Recursive Merge Sort and then calculate the time complexity for it. (20%)

8. Please write down every sequence of each step while finding the Shortest Path from Boston to All

Destinations. (15%)


Boston
4
1500
250
Chicago
3
San
Francisco
Denver
1200
1000
800
1
2
5
New
York
300
1000
1400
New Orleans
900
1700
0
7
1000
6
Miami
Los Angeles