國立臺北大學112學年度日間學士班轉學生招生考試試題

學制系級：資訊工程學系日間學士班3年級

科　　目：資料結構

☐可　☑不可使用計算機

1. (1) Use Heap Sort to sort: 60 80 45 2 36 98 11 47 52 89 75 63 7. Show the sorting process. (5%)
   (2) How does system stack work? (5%)

2. Convert inorder statement: A*B+C*(D-E)/(F+H) to postorder, and print the binary tree. Please convert by the statement order from left to right to generate the unique answer. Also, design an algorithm based on stack to implement the inorder-to-postorder conversion. (15%)

3. Describe the design of using circular linked list for polynomial add, including: (10%)
   (1) add each polynomial by exponential order
   (2) the control of processing values in the linked lists
   (3) The ending condition of completing the adding process

4. Use linked list to implement a stack in C: (10%)

```
typedef struct node { int value; node *next;}
void push (node **top, int new_value) { // assume top is not null
   node *new_node = (node *) malloc (sizeof (node));
   // add new_node to stack by top
}
int pop (node **top) { // assume top is not null
   int value;
   node *deleting_node = *top;
   // remove the deleting node from stack
   free(deleting_node);
 return value;
}
```

5. Use queue to implement level order traversal conversion in C: (15%)

```
void level_order   (tree *root) { //assume root is not null
   node *front, *rear;
   // visit tree nodes based on level order by using queue
   // use queue by front, rear, enqueue(&rear, root), root =dequeue(&front)
   // print the values of tree nodes
   // describe the ending condition
}
```

國立臺北大學112學年度日間學士班轉學生招生考試試題

學制系級：資訊工程學系日間學士班3年級

科　　目：資料結構

☐可　☑不可使用計算機

6. Implement Sparse Matrix Transpose function without nested for loop.(15%)

```
void fast_transpose(term a[ ], term b[ ])
{
   int row_terms[20], starting_pos[20], i, j;
   int num_cols = a[0].col, num_terms = a[0].value;
   b[0].row = num_cols; b[0].col = a[0].row; b[0].value = num_terms; starting_pos[0]=1; bzero(row_terms);
   // Set row_terms[] value
   // Set starting_pos[] value
   // Set b[] value
}
```

7. Maze path search function by stack. (25%)

```
void path()
{    // assume int maze[30][30] and offsets move[8] has been set in main function
     // typedef struct {int v; int h;} offsets;
     int i, row, col, next_row, next_col, dir, found = 0, top =0, mark[30][30];
     typedef struct { int row; int col; int dir;} element
     element position, stack[100];
     mark[1][1] = 1; stack[0].row = 1; stack[0].col = 1; stack[0].dir = 0;
     // Use stack to get current row, col, dir
     // Use dir to calculate next_row, next_col
     // Use next_row, next_col to check this move (1) arrive exit, or
        // (2) become a legal move; add this move into the stack    or
        // (3) check other dir
}
```

試題隨卷繳交