เลขที่นั่งสอบ

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

ข้อสอบปลายภาคการศึกษาที่ 2/2550

วันพุธที่ 5 ~~มกราคม~~ 2551 เวลา 09.00 - 12.00 น.

วิชา CPE 130 Algorithms and Data Structures. น.ศ. วศ.คอมพิวเตอร์ชั้นปีที่ 1A,B

คำสั่ง

1. ข้อสอบมีทั้งสิ้น 7 ข้อ จำนวน 8 แผ่น(รวมแผ่นนี้) ทำทุกข้อลงในข้อสอบที่เว้นช่องไว้ให้
2. ไม่อนุญาตให้นำเครื่องคำนวณใดๆเข้าห้องสอบ
3. ห้ามนำเอกสารใดๆ เข้าห้องสอบ
4. เขียนชื่อ และ รหัสประจำตัว ลงในกระดาษคำตอบทุกแผ่น

.....

(อ.พิพัฒน์ ศุภศิริสันต์)

ผู้ออกข้อสอบ

ข้อสอบนี้ได้ผ่านการประเมินจากภาควิชาวิศวกรรมคอมพิวเตอร์แล้ว

ชื่อ .....รหัสประจำตัว.....ภาควิชา/ชั้นปี.....

ชื่อ ..... รหัสประจำตัว.....เลขที่นั่งสอบ.....

1. กำหนดให้ สมมุติให้ Function สร้างไว้แล้วโดยมีรูปแบบของ Function Prototype ดังนี้ (10 points)

void Push_Stack (double stack[], int *stacktop , double data);

ทำหน้าที่นำค่าที่อยู่ในพารามิเตอร์ data เก็บลงใน stack

double Pop_Stack (double stack[], int *stacktop);

ทำหน้าที่ return ค่าที่อยู่บนสุดของ stack

int Get_Token_Type (char Postfix[], double *data);

ทำหน้าที่ดึงข้อมูลออกจาก string ของ Postfix form ทีละชุดโดยมีการ return ค่าดังนี้

- ถ้าข้อมูลหมดแล้วจะ return ค่าเป็น -1
- ถ้าข้อมูลที่ดึงมาได้เป็นตัวเลขจำนวนจริง จะ return ค่าเป็น 0 และค่าตัวเลขจำนวนจริงที่หาได้จะถูกส่งกลับผ่าน argument ที่ชื่อ data
- ถ้าเป็น operator จะ return เป็นตัวเลขจำนวนเต็มบวก ขึ้นอยู่กับชนิดของ operator

double Calculate (double stack[], int *stacktop, int token_type);

ทำหน้าที่ดึงตัวเลขออกจาก stack แล้วคำนวณหาคำตอบตามชนิดของ Token type แล้วจึงreturn คำตอบที่คำนวณได้กลับไป

จงสร้างฟังก์ชัน ที่ return คำตอบเป็นตัวเลขของการทำ Postfix Polish Notation (Postfix Calculation) ของสมการที่อยู่ในรูปของ postfix form และเก็บอยู่ในตัวแปร Postfix นี้ให้สมบูรณ์

```
double   Postfix_Calculation (char Postfix[])
{ double stack[50] , data,  ans ;
  int  stacktop = 0, token_type ;

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

      ans =  Pop_Stack (stack, &stacktop );
      return  ans;
}
```

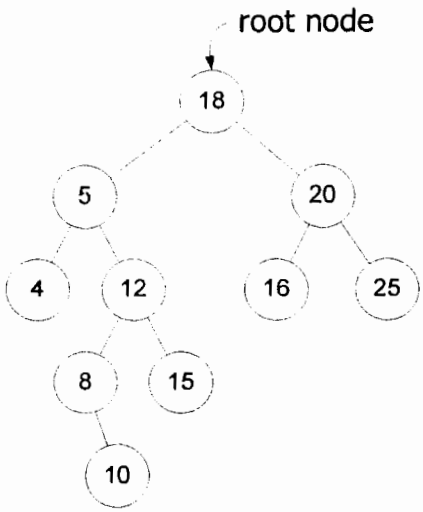
2. สมมุติให้ข้อมูลถูกสร้างไว้แล้วในลักษณะของ Binary Tree โดยมี pointer ชื่อ root ชี้ไปยัง root node ของข้อมูลชุดนี้ โดยมีการกำหนดโครงสร้างข้อมูล ดังนี้ (6 points)

```
typedef  struct  node_tag{ long int id ;
                           char  firstname[15], lastname[20];
                           double  gpa;
                           struct  node_tag  *left, *right; }  node_type;
```

จงสร้างฟังก์ชันสำหรับค้นหา และแสดงผลข้อมูลทั้งหมดที่อยู่ในต้นไม้ที่มีโครงสร้างดังกล่าว โดยฟังก์ชันจะรับค่า root node มา แล้วใช้ขบวนการค้นข้อมูลแบบ Depth First Search ชนิด Pre-order แสดงผล node ละ 1 บรรทัด ประกอบด้วย รหัสประจำตัว ชื่อ นามสกุล และคะแนนเฉลี่ยสะสม)

```
void  Dept_First_Search( node_type  *node)
{
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
}
```

3) กำหนดโครงสร้างของต้นไม้ เป็น ดังรูป (4 points)



จงแสดงต้นไม้ในแต่ละครั้งของการหมุน เพื่อหมุนต้นไม้นี้ให้สมดุล (AVL-Tree)

4) จงเติมฟังก์ชัน sift_down ให้สมบูรณ์ เพื่อใช้เรียงลำดับตัวเลขจำนวนเต็ม ที่เก็บอยู่ในอาร์เรย์ data[] ที่ใช้เทคนิคของ Heap Sort (8 points)

```
void swap(long int *x,long int *y)
{ long int  z ;
  z  =  *x ;  *x  = *y ;  *y = z ;
}

void sift_down(long int data[],  int i, int n)
{ int j , k;
  k = i;
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
}

void  Heap_Sort (long int data[], int n )
{ int i ;
    for (i = n/2; i >=1; i--)                /* Make Heap */
        sift_down(data, i, n);
    for (i = n; i > 1; i--)                    /* Delete root */
      { swap(&data[i], &data[1]);       /* Sort data[i] */
          sift_down(data,1,i-1); }
}
```

5. กำหนดให้ข้อมูลตัวเลขจำนวนเต็ม เก็บอยู่ในอาร์เรย์ int data[1000] ;

- สูตรในการหาค่าตำแหน่งที่เก็บข้อมูลคือ $h(k) = k \bmod 999$
- ฟังก์ชันในการจัดการเมื่อเกิดการชนใช้เป็นแบบ Quadratic Probling คือ $h(k)_i = h(k) + i^2$
- อาร์เรย์ในตำแหน่งที่ไม่มีข้อมูลเก็บอยู่ ค่าของข้อมูลในตำแหน่งนั้นจะเป็น 0

จงสร้างฟังก์ชันที่ใช้ค้นหาข้อมูลเท่ากับตัวแปร k โดย return ตำแหน่งข้อมูลที่ค้นเจอ (0 - 999) ส่วนในกรณีที่ค้นไม่เจอให้ return เป็น -1 (8 points)

ข้อควรระวัง ให้คิดถึงกรณีที่ค่า $h(k)_i$ ที่คำนวณได้มีขนาดใหญ่กว่าขนาดของตารางด้วย

```
int  Hashing (int  data[ ], int  k)
{  int  h , i ;
       i  =  0 ;
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
       if (data[h] == 0)
          return -1 ;
       else  return  h ;
}
```

6. กำหนดให้ ข้อมูลใน Matrix MPK[ ][ ] ใช้แทนกราฟซึ่งแสดงระยะทางระหว่างโหนดแต่ละโหนด เริ่มต้นที่โหนดตำแหน่ง M[0][0] จนถึง M[n-1][n-1] (จำนวน n โหนด) โดยที่ตำแหน่งที่ไม่มีเส้นทาง จะมีค่าเป็น 0 จงใช้ Floyd'Algorithm เพื่อวนรอบคำนวณหา ให้ MPK กลายเป็น Matrix ของระยะทางที่สั้นที่สุดระหว่างโหนดแต่ละโหนด (8 points)

```
void  Floyd (double MPK[ ][100], int  n)
{  double MPL [100][100];
    int  i , j , k , m ;
   for (k = 2; k<= n-1; k++)
        {  for (i=0; i < n; i++)
             for (j = 0; j < n; j++)
                    MPL[i][j] = MPK[i][j];
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

```
        }
}
```

ชื่อ ..... รหัสประจำตัว.....เลขที่นั่งสอบ.....

7. จงสร้างต้นไม้สำหรับรหัสตัวอักษรชุดใหม่ โดยใช้เทคนิคของฮัฟแมนน์ ของข้อความต่อไปนี้ (□ หมายถึงเว้นวรรค) (6 points)

นกกางเขนดีใจ□บินไปบินมา□บินตีลังกา□บินมาบินไป

น	ก	า	ง	เ	ข	ด	ี	ใ	จ	บ	ิ	ไ	ป	ม	ต	ล	ั	□
7	3	4	2	1	1	1	2	1	1	5	5	2	2	2	1	1	1	3