Fundamental Programming I

Labs v1.1

CONTENTS

[This document should be used in conjunction with PPS]

Lab00 - Preparation

Hello World

จงเขียนโปรแกรมเพื่อแสดงข้อความ Hello World บนหน้าจอ

Code Template

```
//main.c
#include<stdio.h>

int main(){

  return 0;
}
```

Test Input

-

Test Output

```
Hello World
...
```

Addition

จงเขียนโปรแกรม เพื่อรับตัวเลขจำนวนเต็ม 2 ตัว จากผู้ใช้ แล้วแสดงผลการบวกออกทางหน้าจอ
โดยโปรแกรมจะมีลักษณะการแสดงผลดังนี้

```
A: <5>
B: <3>
= 8
```

หมายเหตุ ข้อความที่แสดงบนหน้าจอ แล้วมีเครื่องหมาย <> หมายถึง ข้อมูลที่ผู้ใช้พิมพ์

Code Template

```
//main.c
#include<stdio.h>

int main(){
  printf("A:");
  printf("B:");
  return 0;
}
```

Test Input

```
14
-3
```

Test Output

```
A:B:=11
...
```

English Letter

รับตัวอักษรหนึ่งตัว หากเป็นตัวหนังสือ a-z หรือ A-Z ให้แสดงว่า เป็นตัวหนังสือลำดับที่เท่าใดในภาษาอังกฤษ โดยโปรแกรมจะวนรับค่า จนกว่าผู้ใช้จะใส่ตัวหนังสือที่ไม่ใช่ตัวหนังสือภาษาอังกฤษ

รูปแบบการแสดงผล

```
Char: <B>
= 2
Char: <z>
=26
Char: <*>
End.
```

ข้อสังเกต ไม่ว่าผู้ใช้จะใส่ตัวหนังสือเป็น B หรือ b ลำดับตัวอักษรในภาษาอังกฤษจะเป็น 2

หมายเหตุ ห้ามใช้ฟังก์ชันอื่นใดในการประมวลตัวอักษร นอกจาก `printf, scanf`

Code Template

```
//main.c
#include<stdio.h>

int main(){
    printf("Char: ");

    printf("End.\n");
    return 0;
}
```

Test Input

```
c
C
g
%
```

Test Output

```
Char:=3
Char:=3
Char:=7
Char:End.
...
```

Number Is Fun

เกมจะเริ่มต้นทำงานโดยการรับค่าตัวเลขหนึ่งตัวระหว่าง 0-99 หลังจากนั้นจะให้ผู้เล่นทายตัวเลข โดยเกมจะตรวจสอบว่า

- ถ้าตัวเลข ถูกต้อง จบเกม
- ถ้าตัวเลข น้อยกว่า เลขที่กำหนด บอกว่า น้อยไป
- ถ้าตัวเลข มากกว่า เลขที่กำหนด บอกว่า มากไป
- ถ้าตัวเลข มีค่าใกล้เคียงตัวเลขที่กำหนด +3 หรือ -3 ให้บอกว่า ใกล้เคียงแล้ว (โดยไม่บอกว่า มากไปหรือน้อยไป)

โดยผู้เล่นสามารถทายตัวเลขได้เพียง 10 ครั้ง

รูปแบบการแสดงผล

```
T01: <10>
=Too small
T02: <80>
=Too large
T03: <40>
=Too large
T04: <27>
=Very close
T05: <25>
=Victory
```

หมายเหตุ หากผู้เล่นทายครบ 10 ครั้งแล้วยังไม่สามารถทายตัวเลขได้ถูกต้อง ให้ขึ้นความว่า =Game over

Code Template

```
//main.c
#include<stdio.h>

int main(){
    int i;
    printf("Ans: ");

    for(i = 0; i < 100; i++)
       printf("\n");

    printf("T%02d: ");
    printf("=Too small\n");
    printf("=Too large\n");
    printf("=Very close\n");
    printf("=Victory\n");
    printf("=Game over\n");

    return 0;
}
```

Test Input

```
50
20
60
40
45
52
48
50
```

Test Output

```
T01:=Too small
T02:=Too large
T03:=Too small
T04:=Too small
T05:=Very close
T06:=Very close
T07:=Victory
```

Encryption

เขียนโปรแกรมเพื่อรับตัวอักษรโดยใช้ฟังก์ชัน `getchar()` จนกว่าจะเจอ `EOF`

ให้แสดงผลตัวหนังสือที่รับเข้าไปในลักษณะของการเข้ารหัส โดยมีการทำงานดังนี้

- หากเป็นตัวหนังสือ a-z ให้เลื่อนตัวอักษรไปข้างหน้า 2 ตัว
- หากเป็นตัวหนังสือ A-Z ให้เลื่อนตัวอักษรไปข้างหลัง 1 ตัว
- หากเป็นอักขระอื่นๆ ให้แสดงตัวอักษรนั้นได้เลย

โดยโปรแกรมจะต้องใช้ฟังก์ชันข้างล่าง

```
char enc_char(char c);
```

เลื่อนตัวหนังสือตามเงื่อนไขการเข้ารหัส

```
int check_char(char c);
```

ตรวจสอบว่าตัวหนังสือเป็นตัวพิมพ์เล็กหรือตัวพิมพ์ใหญ่ โดยคืนค่าเป็น 0 หากเป็นตัวอักษรตัวเล็ก 1 หากเป็นตัวอักษรตัวใหญ่ และ -1 หากเป็นตัวอักขระอื่นๆ

รูปแบบการแสดงผล

```
<HelLo * WorlD!>
GgnKq * VqtnC!
<EOF>
```

หมายเหตุ EOF

บน Linux, Mac กด `Ctrl+d` ได้ทันที

บน Windows กด `Ctrl+z` ในบรรทัดว่างๆ แล้วกด Enter

Code Template

```
//main.c
#include<stdio.h>
char enc_char(char c);
int check_char(char c);

int main(){

    return 0;
}

char enc_char(char c){
    int cs = check_char(c);

}

int check_char(char c){

}
```

Test Input

```
FoOBar SaZb!&
z*
```

Test Output

```
EqNAct RcYd!&
b*
...
```

Pointer-101

จงเขียนข้อความรับตัวอักษรภาษาอังกฤษ a-z จำนวนหนึ่งตัว (ตัวอักษรตัวเล็กเท่านั้น) แล้วแสดงตัวอักษรถัดไป โดยตัวอักษรถัดไปของ z คือ a

โดยการหาตัวอักษรถัดไป ให้ใช้ฟังก์ชัน

```
void to_next(char* c);
```

รูปแบบการแสดงผล

```
Enter: <g>
= h
```

Code Template

```
//main.c
#include<stdio.h>

void to_next(char* cptr);

int main(){
  char c;
  printf("Enter: ");

  printf("=");
  return 0;
}
```

Test Input

```
z
```

Test Output

```
Enter:=a
...
```

Next Char

จงรับข้อความหนึ่งข้อความ และตัวอักษรหนึ่งตัว โปรแกรมจะหาว่า ในข้อความที่กำหนด ตัวอักษรถัดไปจากตัวอักษรที่กำหนดคือตัวอะไร หากไม่พบหรือตัวถัดไปเป็นตัวสิ้นสุดข้อความ ให้แสดงเฉพาะเครื่องหมาย = เป็นคำตอบ

รูปแบบการแสดงผล

```
S: Hello world
C: e
= l
```

ทั้งนี้จะต้องมีการใช้ฟังก์ชัน `find_char` อย่างเหมาะสม

```
char* find_char(char s[], int c);
```

คืนค่าพอยเตอร์ที่ชี้ไปยังตัวอักษร c ภายในข้อความ s แต่หากไม่พบตัวอักษรดังกล่าว จะคืนค่าเป็น NULL

Code Template

```
//main.c
#include<stdio.h>

char* find_char(char s[], int c);

int main(){
  char str[128];
  char c;

  printf("S:");
  printf("C:");
  printf("=");

  return 0;
}
```

Test Input

```
Enter me
m
```

Test Output

```
S:C:=e
...
```

Sum with Pointer

รับตัวเลขจำนวนเต็ม 5 ตัว แล้วหาผลบวกของสมาชิกทุกตัว ผ่านฟังก์ชัน `sum`

```
int sum(int *p, int *q);
```

โดยห้ามใช้เครื่องหมาย [] ในฟังก์ชัน `sum` แต่ให้ใช้ประโยชน์จากการเพิ่มค่าของพอยเตอร์

Code Template

```
//main.c
#include<stdio.h>
#define NUM 5

int sum(int *p, int *q);

int main(){
    int i, a[NUM];
    for(i = 0; i < NUM; i++){
        printf("N%02d:", i + 1);
        scanf("%d", &a[i]);
    }
    printf("=%d\n", sum(&a[0], &a[NUM]));
    return 0;
}
```

Test Input

```
3
1
2
0
4
```

Test Output

```
N01: N02: N03: N04: N05: = 10
...
```

Max to Sort

รับค่าจำนวนเต็ม จำนวน 5 ตัว เก็บไว้ในอาเรย์ หลังจากนั้นใช้ประโยชน์จากฟังก์ชัน `find_max` เพื่อพิมพ์ข้อมูลเรียงจากมากไปหาน้อย โดยมีสมมุติฐานดังต่อไปนี้

1) ข้อมูลที่อยู่ในอาเรย์มากกว่าหรือเท่ากับ 0
2) สามารถเปลี่ยนแปลงข้อมูลในอาเรย์ได้

ทั้งนี้ฟังก์ชัน `find_max` มีต้นแบบของฟังก์ชันดังนี้

```
int* find_max(int *p, int *q);
```

=> `p` จุดเริ่มต้นของอาเรย์

=> `q` จุดสิ้นสุดของอาเรย์

=> คืนค่า เป็น พอยเตอร์อ้างอิงไปยังสมาชิกที่มีค่ามากที่สุด

รูปแบบการแสดงผล

```
N01: <3>
N02: <1>
N03: <2>
N04: <5>
N05: <7>
= 7 5 3 2 1
```

Code Template

```
//main.c
#include<stdio.h>
#define LEN 5

int* find_max(int *p, int *q);

int main(){
  int d[LEN], i;

  for(i = 0; i < LEN; i++){
    printf("N%02d: ", i + 1);
    scanf("%d", &d[i]);
  }

  return 0;
}
```

Test Input

```
3
2
0
7
5
```

Test Output

```
N01:N02:N03:N04:N05:=7 5 3 2 0
...
```

Another Find Max

รับค่าจำนวนเต็ม จำนวน 5 ตัว เก็บไว้ในอาเรย์ หลังจากนั้นใช้ประโยชน์จากฟังก์ชัน `max` เพื่อหาค่ามากที่สุด
ทั้งนี้ฟังก์ชัน `max` มีต้นแบบของฟังก์ชันดังนี้

```
int* max(int *a, int *b);
```

=> `a` ตำแหน่งของตัวแปรตัวที่ 1

=> `b` ตำแหน่งของตัวแปรตัวที่ 2

=> คืนค่า เป็น พอยเตอร์อ้างอิงไปยังตัวแปรที่มีค่ามากกว่า

ข้อสังเกต ฟังก์ชันนี้ประมวลผลกับตัวแปรเดี่ยว ไม่ใช่อาเรย์

รูปแบบการแสดงผล

```
N01: <3>
N02: <1>
N03: <2>
N04: <5>
N05: <7>
= 7
```

Code Template

```
//main.c
#include<stdio.h>
#define LEN 5

int* max(int *a, int *b);

int main(){
  int d[LEN], i;

  for(i = 0; i < LEN; i++){
    printf("N%02d: ", i + 1);
    scanf("%d", &d[i]);
  }

  return 0;
}
```

Test Input

```
3
2
0
7
5
```

Test Output

```
N01:N02:N03:N04:N05:=7
```

Replace Space

เขียนโปรแกรมเพื่อรับข้อความหนึ่งบรรทัด โดยใช้ประโยชน์จากฟังก์ชัน

```
void replace_space(char *s, char c);
```

เพื่อแทนที่ space ด้วยตัวอักษร c

รูปแบบการแสดงผล

```
Enter: <I love you>
Char: _
= I_love_you
```

Code Template

```
//main.c
#include<stdio.h>

void replace_space(char *s, char c);

int main(){
    printf("Enter: ");
    printf("Char: ");
    printf("= ");

    return 0;
}

void replace_space(char *s, char c){

}
```

Test Input

```
Hello world na ja.
*
```

Test Output

```
Enter:Char:=Hello*world*na*ja.
```

Replace_space2

เขียนโปรแกรมเพื่อรับข้อความหนึ่งบรรทัด โดยใช้ประโยชน์จากฟังก์ชัน

```
char* find_space(char *s);
```

ซึ่งจะทำหน้าที่ ในการคืนค่าพอยเตอร์ไปยังตัวอักษรที่เป็น space และจะคืนค่า NULL หากในข้อความนั้นไม่มี space เพื่อแทนที่ space ด้วยตัวอักษรที่กำหนดจากผู้ใช้

รูปแบบการแสดงผล

```
Enter: <I love you>
Char: _
= I_love_you
```

Code Template

```
//main.c
#include<stdio.h>

char* find_space(char *s);

int main(){
    printf("Enter: ");
    printf("Char: ");
    printf("= ");

    return 0;
}


char* find_space(char *s){

}
```

Test Input

```
Hello world na ja.
*
```

Test Output

```
Enter:Char:=Hello*world*na*ja.
```

V Processing

เขียนโปรแกรม เพื่อรับข้อมูลเข้าสู่ `struct v` แล้วประมวลผลหาค่า f ที่ต้องการตามสูตรที่กำหนดให้

```
f = v.k + (v.set.factor * sum(v.set.values))
```

`v.k` คือ ค่า `k` ของ `v`

`v.set.factor` คือ ค่า factor ของ set ของ `v`

`sum(v.set.values)` คือ ผลรวมของ element ทั้งหมดในอาเรย์ values ของ set ของ `v` ซึ่งจะมีจำนวนตามที่ผู้ใช้ระบุ

รูปแบบการแสดงผล

```
k = <12>
set.factor = <1.5>
set.amount = <5>
set.values = <3 1 4 5 7>
f = 42.0
```

Code Template

```
//main.c
#include<stdio.h>
#include<stdlib.h>

struct set{
    float factor;
    int amount;
    int *values;
};
typedef struct set SET;

struct v{
    int k;
    SET set;
};
typedef struct v V;

void input_v(V *p);
int sum(int d[], int len);
float f_v(V *p);


int main(){
    V v;
    input_v(&v);

    printf("f = %.1f", f_v(&v));

    //free values here

    return 0;
}
```

```
void input_v(V *p){
    printf("k = ");
    printf("set.factor = ");
    printf("set.amount = ");

    //malloc your values here

    printf("set.values = ");
}

int sum(int d[], int len){

}

float f_v(V *p){

}
```

Test Input

```
14
1.5
5
3 1 4 5 7
44
```

Test Output

```
k = set.factor = set.amount = set.values = f = 44.0
```

Simple Text File

ไฟล์ num.txt ในแต่ละบรรทัดจะมีตัวเลขจำนวนเต็ม 1 ตัวเลข จงเขียนโปรแกรมเพื่อหาผลรวมของตัวเลข

ตัวอย่างไฟล์ num.txt

```
7
2
1
```

รูปแบบการแสดงผล

```
=10
```

หากมีการแก้ไขไฟล์ num.txt อาจทำให้ `testing failed.`

Code Template

```
//main.c
#include<stdio.h>

int main(){
  FILE *fp;


  return 0;
}
```

Code Template

```
num.txt
9
2
3
6
4
1
7
21
0
5
```

Test Input

-

Test Output

```
=58
```

Array of Employee

จงเขียนโปรแกรมที่ทำการอ่านไฟล์ไบนารี ชื่อ `employee.bin`
โดยไฟล์จะเริ่มต้นจากตัวเลขจำนวนเต็ม บอกจำนวนข้อมูลพนักงานที่บันทึกไว้ในไฟล์ จากนั้นประมวลผลข้อมูลให้อยู่ในรูปของ `struct employee` ที่กำหนดให้ แล้วทำการแสดงข้อมูลของพนักงานทุกคน พร้อมแสดงผลเงินเดือนรวม
โดยการอ่าน จะเป็นการอ่านเข้าสู่อาเรย์อย่างเหมาะสม - ข้อนี้มีการใช้งาน `malloc`
จะต้องมีการกด L-test อย่างน้อยหนึ่งครั้ง ก่อนการใช้ L-Execute เพื่อดาว์นโหลดไฟล์ที่เกี่ยวข้องโดยอัตโนมัติ
รูปแบบการแสดงผล

```
John Doe:3000.0
Mark Ken:2300.0
Sucy Merc:2000.0
=7300.0
```

Code Template

```
//main.c
#include <stdio.h>

struct employee {
  char name[128];
  float salary;
};

typedef struct employee Employee;
float total_salary(Employee all[], int len);

int main() {
  int num; //number of employees
  Employee *all;
  FILE *fp;

  printf("=%.1f\n", total_salary(all, num));
  return 0;
}
```

Test Input

-

Test Output

```
John Doe:3000.0
Mark Ken:2300.0
Sucy Merc:2000.0
=7300.0
```

Insert At Front

รับข้อมูลตัวเลขจำนวนเต็ม จนกว่าผู้ใช้จะใส่ข้อมูลที่น้อยกว่าหรือเท่ากับ 0 นำข้อมูลใส่ linked list ในรูปแบบ "Insert at front" และทำการแสดงผลข้อมูล พร้อมหาค่าผลรวมของข้อมูลทั้งหมด

```
Enter: <1 3 2 7 4 -1>
= 4 7 2 3 1
= 17
```

Code Template

```
//main.c
#include<stdio.h>

struct listnode {
  int data;
  struct listnode *next;
};

typedef struct listnode LN;

void insert_at_front(LN **hptr, int d);
void print(LN *head);
int sum(LN *head);

int main(){
  LN *head;
  int d;

  printf("Enter: ");
  do{
    scanf("%d", &d);
    if(d > 0){

    }
  }while(d > 0);

  printf("=");
  print(head);
  printf("\n=%d", sum(head));

  return 0;
}

void insert_at_front(LN **hptr, int d){

}
```

```
void print(LN *head){

}

int sum(LN *head){

}
```

Test Input

```
1 5 3 2 7 -1
```

Test Output

```
Enter: = 7 2 3 5 1
= 18
...
```

Insert At Back

รับข้อมูลตัวเลขจำนวนเต็ม จนกว่าผู้ใช้จะใส่ข้อมูลที่น้อยกว่าหรือเท่ากับ 0 นำข้อมูลใส่ linked list ในรูปแบบ "Insert at back" และทำการแสดงผลข้อมูล พร้อมหาค่าผลรวมของข้อมูลทั้งหมด

```
Enter: <1 3 2 7 4 -1>
= 1 3 2 7 4
= 17
```

สามารถนิยามฟังก์ชันเพิ่มเติมได้

Code Template

```
//main.c
#include<stdio.h>

struct listnode {
  int data;
  struct listnode *next;
};

typedef struct listnode LN;

void insert_at_back(LN **hptr, int d);
void print(LN *head);
int sum(LN *head);
```

```
int main(){
  LN *head;
  int d;

  printf("Enter: ");
  do{
    scanf("%d", &d);
    if(d > 0){

    }
  }while(d > 0);

  printf("=");
  print(head);
  printf("\n=%d", sum(head));

  return 0;
}
void insert_at_front(LN **hptr, int d){

}

void print(LN *head){

}

int sum(LN *head){

}
```

Test Input

```
1 5 3 2 7 -1
```

Test Output

```
Enter: = 1 5 3 2 7
= 18
```

Lab07 – More Linked List

Linked Song 1

รับข้อมูล Song จากผู้ใช้จำนวน 5 เพลง เก็บไว้ในลิงค์ลิสต์ โดยใช้เทคนิค insert at back โดยให้ถือว่าชื่อเพลงไม่มีเว้นวรรค เขียนโค้ดโดยใช้ต้นแบบของฟังก์ชันที่ให้มาอย่างเหมาะสม

```
S01: <I_Love_You 180.4>
...
= 3120.50
```

Code Template

```
//main.c
#include<stdio.h>
struct song{
  char title[128];
  double duration;
};
typedef struct song Song;

struct listnode{
  Song s;
  struct listnode *next;
};
typedef struct listnode LN;

void insert(LN **hptr, Song *sp);
LN *find_tail(LN *head);
double sum_duration(LN *head);

int main(){
  Song buf;

  printf("=%.2f");
  return 0;
}

void insert(LN **hptr, Song *sp){

}

LN *find_tail(LN *head){

}

double sum_duration(LN *head){

}
```

Test Input

```
A 300
B 100
C 200
D 150
E 50
```

Test Output

```
S01: S02: S03: S04: S05: = 800.00
```

Linked Song 2

รับข้อมูล Song จากผู้ใช้จำนวน 5 เพลง เก็บไว้ในลิงค์ลิสต์ โดยใช้เทคนิค insert at back โดยให้ถือว่าชื่อเพลงไม่มีเว้นวรรค เขียนโค้ดโดยใช้ต้นแบบของฟังก์ชันที่ให้มาอย่างเหมาะสม

```
S01: <I_Love_You 180.4>
...
= 3120.50
```

Code Template

```
//main.c
#include<stdio.h>
struct song{
  char title[128];
  double duration;
};
typedef struct song Song;

struct listnode{
  Song *sp;
  struct listnode *next;
};
typedef struct listnode LN;

Song *input_song();
void insert(LN **hptr, Song *sp);
LN *find_tail(LN *head);
double sum_duration(LN *head);

int main(){
  Song *sp;
  int i;
  for(i = 0; i < 5; i++){
    sp = input_song();  //malloc, scanf Song in this function
  }

  printf("=%.2f");
  return 0;
}
```

```
Song *input_song(){

}

void insert(LN **hptr, Song *sp){

}

LN *find_tail(LN *head){

}

double sum_duration(LN *head){

}
```

Test Input

```
A 300
B 100
C 200
D 150
E 50
```

Test Output

```
S01: S02: S03: S04: S05: = 800.00
```

Lab08 - Stack

Postfix-notation

จงรับข้อความซึ่งเป็นการเขียนนิพจน์ทางคณิตศาสตร์แบบ Postfix โดยให้ตั้งสมมติฐานว่า นิพจน์ดังกล่าว จะประกอบด้วยตัวเลขเฉพาะหลักหน่วย และเครื่องหมาย บวก ลบ และคูณเท่านั้น แต่อาจมีเว้นวรรคหรือไม่เว้นวรรคก็ได้
หลังจากนั้นทำการคำนวณหาค่าผลลัพธ์โดยใช้ stack ในการประมวลผล
รูปแบบการแสดงผล

```
Enter: <921+->
= 6
```

หมายเหตุ อาจใส่ข้อมูลเป็น 9 2 1 + - ก็ได้
**อนุญาตให้ใช้ array แทน linked-list ได้

Code Template

```
//main.c
#include<stdio.h>

struct listnode{
   int data;
   struct listnode *next;
};

typedef struct listnode LN;
typedef LN *LNP;

typedef struct{
   int size;
   LNP head;
} Stack;

void push(Stack *sptr, int v);
int pop(Stack *sptr);
void cal(Stack *sptr, char operator);

int main(){
   Stack stack = {0, NULL};
   char s[128]; char *tmp;
   printf("Enter: "); gets(s);

   return 0;
}
```

Test Input

```
231 +- 4*
```

Test Output

```
Enter:= -8
```

Perfect Tree

จงรับข้อมูลจำนวนบวกจากผู้ใช้ จนกว่าผู้ใช้จะใส่ 0 หรือ เลขติดลบ นำตัวเลขที่ได้ไปสร้าง tree โดยใช้ฟังก์ชัน `insert_node` ตามลำดับการป้อนข้อมูลของผู้ใช้

นิยามและเรียกใช้ฟังก์ชัน `is_perfect_tree` เพื่อตรวจสอบว่า tree ดังกล่าวสมดุลอย่างสมบูรณ์แบบหรือไม่

โดย tree จะสมดุลอย่างสมบูรณ์แบบ เมื่อ size ด้านซ้าย เท่ากับจำนวน size ด้านขวา สำหรับทุกๆ sub-tree

รูปแบบการแสดงผล

```
N01: <5>
N02: <2>
N03: <7>
N04: <1>
N05: <3>
N06: <6>
N07: <8>
N08: <0>
=Yes
```

Code Template

```c
//main.c
#include <stdio.h>
#include "integer_tree.h"

int is_perfect_tree(TREE t);

int main(){
   TREE t = NULL;
   int i = 0, tmp;
   do{
      printf("N%02d: ", i + 1);
      scanf(" %d", &tmp);

      i++;
   }while(tmp > 0);

  printf(" = %s\n", is_perfect_tree(t) ? "Yes" : "No");
  return 0;
}
```

Code Template

```
//integer_tree.h
#include <stdio.h>

struct treenode{
	struct treenode *leftptr;
	int data;
	struct treenode *rightptr;
	};
typedef struct treenode TREENODE;
typedef TREENODE *TREE;

void insert_node(TREE *tp, int value) {
	/* tp is a pointer to a BST */
	if (*tp == NULL) {
		*tp = malloc(sizeof(TREENODE));
		(*tp)->data = value;
		(*tp)->leftptr = NULL;
		(*tp)->rightptr = NULL;
	}else if (value < (*tp)->data )
		insert_node(&((*tp)->leftptr), value);
	else if (value > (*tp)->data )
		insert_node(&((*tp)->rightptr), value);
	else
		printf("duplicate node\n");
}
```

Test Input

```
5 2 7 1 3 6 8 0
```

Test Output

```
N01: N02: N03: N04: N05: N06: N07: N08: =Yes
```