PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Final Examination: ภาคการศึกษาที่ 1 **Academic Year:** 2549

Subject Number: 240-204

Subject Title: Data Structures and Computer Programming Techniques

---

ทุจริตในการสอบ มีโทษขั้นต่ำ คือ ปรับตกในรายวิชาที่ทุจริต และพักการเรียน 1 ภาคการศึกษา

อ่านรายละเอียดของข้อสอบ และคำแนะนำให้เข้าใจก่อนเริ่มทำข้อสอบ

รายละเอียดของข้อสอบ:

เวลา 3 ชั่วโมง (180 คะแนน: 180 นาที)
เอกสารมีทั้งหมด 4 หน้า (ไม่รวมหน้านี้) คำถามจำนวน 3 ข้อ

สิ่งที่สามารถนำเข้าห้องสอบได้:

อนุญาต: กระดาษขนาด A4 ที่เขียนด้วยลายมือตนเอง 1 แผ่น และเครื่องเขียนต่างๆ
ไม่อนุญาต: หนังสือ และเครื่องคิดเลข

คำแนะนำ:

- พยายามทำทุกข้อ
- คำตอบทั้งหมดจะต้องเขียนลงในสมุดคำตอบ
- คำตอบส่วนใดอ่านไม่ออก จะถือว่าคำตอบนั้นผิด
- อ่านคำสั่งในแต่ละข้อให้ชัดเจนว่า เขียนโปรแกรมบางส่วน เขียนฟังก์ชัน หรือเขียนทั้งโปรแกรม รวมไปถึงข้อกำหนดเพิ่มเติม และหมายเหตุในข้อนั้นๆ
- การเขียนโปรแกรมในแต่ละข้อ อาจจะไม่ต้องเขียนตามคำสั่งย่อยทั้งหมด แต่คะแนนจะลดลงตามส่วน และหากในข้อใหญ่หนึ่งข้อ นักศึกษาไม่สามารถทำข้อย่อยข้อแรกๆ ได้ นักศึกษาสามารถทำข้อย่อยหลังๆ โดยให้อ้างอิงเหมือนนักศึกษาทำข้อย่อยข้อแรกๆ ได้
- การเขียน code จะต้องตั้งชื่อตัวแปรให้เหมาะสม และมี comment ในจุดสำคัญต่างๆ โดยให้ทั้งหมดเป็นไปตามหลักการเขียนโปรแกรมที่ดี

ข้อที่ 1 WARM UP *(60 คะแนน: 60 นาที)*

1.1 จากโปรแกรมต่อไปนี้ (10 คะแนน)

```
#include <iostream>

using std::cout;
using std::endl;

class MyObject {
   public:
        MyObject(int);
        int data;
};

MyObject::MyObject(int dummy) { data = dummy; }

int foo1(MyObject obj) { return obj.data++;  }

int foo2(MyObject &obj){ return obj.data++;  }

int foo3(MyObject *obj) { return (obj->data)++; }

int main(){
  MyObject a(1);
  cout << "foo1 function output = " << /*call foo1*/ << endl ;
  cout << "foo2 function output = " << /*call foo2*/ << endl ;
  cout << "foo3 function output = " << /*call foo3*/ << endl;
  cout << "After foo function = " << a.data << endl;
  return 0;
}
```

จงเรียกใช้ foo1, foo2 และ foo3 อย่างเหมาะสม พร้อมทั้งแสงผลลัพธ์ของการรันโปรแกรม

1.2 จากข้อมูลทั้ง 10 จำนวนต่อไปนี้ จงวาดต้นไม้แบบ Binary Search Tree

7 5 3 1 4 9 11 10 12 8 2 (10 คะแนน)

พร้อมทั้งเขียนผลการเยี่ยมเยียน node ต่าง ๆ แบบ Pre-order และ In-order และ Post-order ตามลำดับ

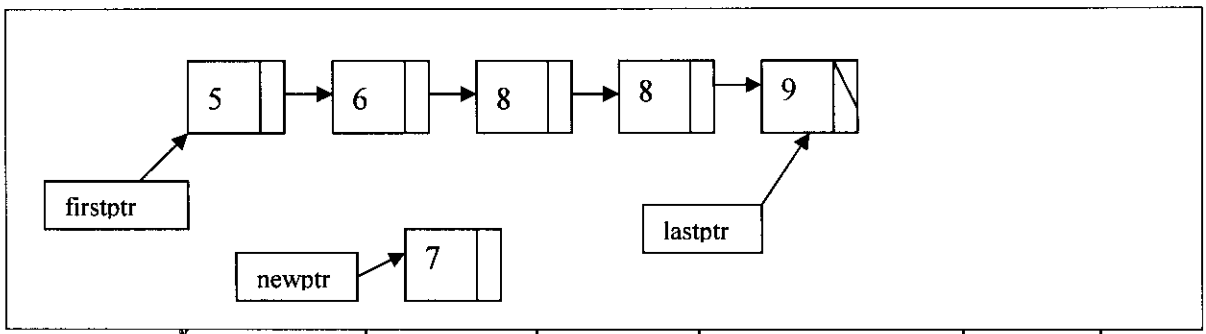
1.3 ในแต่ละหัวข้อย่อย จงระบุว่าเราควรใช้ Container ใดใน STL สำหรับการจัดเก็บข้อมูล

(10 คะแนน)

- ข้อมูลมีเพิ่มขึ้นเรื่อย ๆ
- ข้อมูลมีจำนวนไม่มาก แต่ต้องมีการสลับตำแหน่งบ่อยครั้ง
- ข้อมูลเป็นระดับพนักงานงาน กับอัตราค่าจ้างรายชั่วโมง
- ข้อมูลมีการเปลี่ยนแปลงที่ช่วงต้นและช่วงปลายของข้อมูลบ่อยครั้ง
- ข้อมูลมีจำนวนที่แน่นอน และต้องการอ่านเพียงอย่างเดียว

1.4 หากเราต้องการฟังก์ชัน push_back ของ vector ไม่อนุญาตให้มีการใส่ออบเจ็คซ้ำกันเข้าไปใน vector จงอธิบายว่าเทคนิค Inheritance และ Polimorphism จะเข้ามาช่วยอย่างไร (ควรมีโค้ดสั้น ๆ ประกอบคำอธิบาย) (10 คะแนน)

1.5 ในข้อนี้ต้องการให้สร้าง Linked List ที่มีการเรียงข้อมูลจากน้อยไปมาก (ascending order) ดังรูป



จงวาดรูปทีละขั้นตอนของการเลื่อน pointer เพื่อแทรกโหนดที่มีเลข 7 ลงไปใน List ที่กำหนดให้ ซึ่งในการแทรกจะต้องคำนึงถึงการเรียงของข้อมูลด้วย (สามารถใช้ pointer อื่น ๆ เพิ่มเติมได้) (10 คะแนน)

1.6 จงอธิบายถึง 2 แนวทางการนำ Linked List ไปสร้างเป็น Stack (10 คะแนน)

ข้อที่ 2 POLYMORPHISM & STL *(60 คะแนน: 60 นาที)*

จาก `class Shape2D` ที่กำหนดให้ จงเขียนโปรแกรมเพื่อคำนวนหาผลรวมของพื้นที่ของรูปร่างต่าง ๆ โดยให้ใช้รายละเอียดเพิ่มเติมดังต่อไปนี้

```
class Shape2D{
   double getArea();
};
```

2.1 จงปรับปรุงคลาส `Shape2D` อย่างเหมาะสม โดยให้ `Shape2D` ไม่จำเป็นต้องนิยามฟังก์ชัน `getArea` และเหมาะสมสำหรับการใช้ Polymorphism (10 คะแนน)

2.2 จงเขียน `class Rectangle` อย่างสมบูรณ์ ซึ่งประกอบด้วยความกว้างและความยาว โดยให้สืบทอดจาก `Shape2D` อย่างเหมาะสม (10 คะแนน)

2.3 จงเขียน `class Triangle` อย่างสมบูรณ์ ซึ่งประกอบด้วยฐานและส่วนสูง โดยให้สืบทอดจาก `Shape2D` อย่างเหมาะสม (10 คะแนน)

2.4 จงเขียนฟังก์ชัน `main` ที่มีการทำงานดังต่อไปนี้ (20 คะแนน)

- สร้างออบเจ็ค `Rectangle` ซึ่งมีความกว้างและความยาวเป็น 2x7 และ 5x4
- สร้างออบเจ็ค `Triangle` ซึ่งมีฐานและส่วนสูงเป็น 3x6
- นำออบเจ็คทั้งสามตัวไปใส่ใน STL ที่เหมาะสม
- ใช้ `iterator` ในการคำนวนหาพื้นที่รวมของออบเจ็คทั้งหมดใน STL พร้อมทั้งแสดงผล

2.5 หากเราต้องการให้ออบเจ็คที่สร้างจาก Rectangle และ Triangle เก็บชื่อซึ่งเป็น char[] ไม่เกิน 32 ตัวอักษร เราควรปรับปรุงโค้ดอย่างไร (เขียนโค้ดประกอบ) (10 คะแนน)

ข้อที่ 3 DATA STRUCTURE WITH POINTERS *(60 คะแนน: 60 นาที)*

โปรแกรมในข้อนี้จะเป็นการจำลองเกมส์บันไดงู ซึ่งจะมีหมากเพียงตัวเดียว แต่ละตาจะมีการสุ่มตัวเลขตั้งแต่ 1-3 เพื่อให้หมากเดินไปบนบอร์ดตามจำนวนตัวเลขที่สุ่มได้ หากตารางที่ตกนั้นมีบันได (`bridge`) เชื่อมไปยังตารางอื่น ตัวหมากจะเปลี่ยนตำแหน่งไปยังปลายทางของบันได

จากคลาสที่กำหนดให้ และส่วนของฟังก์ชัน `main` ให้นักศึกษาเขียนนิยามของฟังก์ชันต่าง ๆ ของ `class Board` (ที่ยังไม่มีการนิยาม) อย่างเหมาะสม เพื่อให้โปรแกรมทำงานได้อย่างถูกต้องและสอดคล้องกับผลการรันโปรแกรม

`Board(int amount)` จะทำการกำหนดจำนวนตารางบนบอร์ดตามค่าที่ระบุใน `amount`

`~Board()` จะทำการจัดการหน่วยความจำอย่างเหมาะสม

`void link(int pos1, int pos2)` จะทำการเชื่อมสะพานระหว่างตารางลำดับที่ `pos1` กับตารางลำดับที่ `pos2`

`BNode* getNodeByPos(int pos)` จะทำการค้นหา pointer ที่อ้างอิงไปยังตารางลำดับที่ `pos`

`BNode* walk(int turn)` จะทำการเลื่อนตัวหมากไปจำนวน `turn` ช่อง และคืนค่าเป็น pointer ที่อ้างอิงไปยังตารางตำแหน่ง ณ.ที่ ตัวหมากอยู่

`BNode* getCurrentPos()` คืนค่าเป็น pointer ที่อ้างอิงไปยังตารางตำแหน่ง ณ.ที่ ตัวหมากอยู่

`int getTotolMark()` ทุกครั้งที่มีการเดิน คะแนนสะสมจะเพิ่มขึ้นตามหมายเลขตารางที่ตัวหมากเดินไปตก โดยเริ่มต้น `mark` จะเป็น 0

```
class BNode{
  friend class Board;
  private:
    int pos;
    BNode* next;
    BNode* bridge;
  public:
    BNode(int pos):pos(pos){}
    int getPos(){ return pos; }
};

class Board{
  private:
    BNode* head;
    BNode* cur;
    int mark;
  public:
    Board(int amount);
    ~Board();
    void link(int pos1, int pos2);
    BNode* getNodeByPos(int pos);
    BNode* walk(int turn);
    BNode* getCurrentPos() { return cur; }
    int getTotalMark(){ return mark; }
};
```

```
#include<iostream>
#include<stdlib.h>
#include "board.h"

int main()
{
  Board b(15);
  int briges[5][2] = {{1, 5}, {4, 2}, {3, 5}, {10, 2}, {12, 7}};

  for(int i = 0; i < 5; i++)
    b.link(briges[i][0], briges[i][1]);

  int turn = 0;
  srand( time(NULL) );
  while(b.walk(turn = rand()%3 + 1) != NULL){
    std::cout << "You walk " << turn << " turns to " <<
b.getCurrentPos()->getPos() << "\n";
  }
  std::cout << "Your score = " << b.getTotalMark() << "\n";
  return 0;
}
```

Output

```
Crossing Bridge to 5
You walk 2 turns to 5
You walk 3 turns to 8
Crossing Bridge to 2
You walk 2 turns to 2
You walk 3 turns to 5
You walk 2 turns to 7
You walk 1 turns to 8
You walk 3 turns to 11
You walk 2 turns to 13
You walk 1 turns to 14
You walk 1 turns to 15
Your score = 157
```