PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Final Examination: ภาคการศึกษาที่ 2 **Academic Year:** 2548

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 12 หน้า (ไม่รวมหน้านี้) คำถามจำนวน 4 ข้อ

สิ่งที่สามารถนำเข้าห้องสอบได้:

อนุญาต: กระดาษขนาด A4 ที่เขียนด้วยตนเอง 1 แผ่น และเครื่องเขียนต่าง ๆ
ไม่อนุญาต: หนังสือ และเครื่องคิดเลข

คำแนะนำ:

- พยายามทำทุกข้อ
- คำตอบทั้งหมดจะต้องเขียนลงในสมุดคำตอบ
- คำตอบส่วนใดอ่านไม่ออก จะถือว่าคำตอบนั้นผิด
- อ่านคำสั่งในแต่ละข้อให้ชัดเจนว่า เขียนโปรแกรมบางส่วน เขียนฟังก์ชัน หรือเขียนทั้งโปรแกรม รวมไปถึงข้อกำหนดเพิ่มเติม และหมายเหตุในข้อนั้น ๆ
- การเขียนโปรแกรมในแต่ละข้อ อาจจะไม่ต้องเขียนตามคำสั่งย่อยทั้งหมด แต่คะแนนจะลดลงตามส่วน และหากในข้อใหญ่หนึ่งข้อ นักศึกษาไม่สามารถทำข้อย่อยข้อแรก ๆ ได้ นักศึกษาสามารถทำข้อย่อยหลัง ๆ โดยให้อ้างอิงเหมือนนักศึกษาทำข้อย่อยข้อแรก ๆ ได้
- การเขียน code จะต้องตั้งชื่อตัวแปรให้เหมาะสม และมี comment ในจุดสำคัญต่าง ๆ โดยให้ทั้งหมดเป็นไปตามหลักการเขียนโปรแกรมที่ดี

ข้อที่ 1 WARM UP *(30 คะแนน: 30 นาที)*

1.1 จากโปรแกรมต่อไปนี้ จงเขียนผลลัพธ์การทำงานของโปรแกรม (6 คะแนน)

```
#include <iostream>

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

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

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

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

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

int foo2(MyObject *obj) { return ++(obj->data); }

int main(){
  MyObject a = MyObject(3);
  cout << "foo function output = " << foo(a) << endl ;
  cout << "foo1 function output = " << foo1(a) << endl ;
  cout << "foo2 function output = " << foo2(&a) << endl;
  cout << "After foo function = " << a.data << endl;
  return 0;
}
```

1.2 จงอธิบายความหมายของ Abstract class พร้อมยกตัวอย่างประกอบ (4 คะแนน)

1.3 จงอธิบายการเข้าถึงทั้ง 3 อย่าง คือ `public`, `protected`, `private` สำหรับ member ของ class ต่าง ๆ และ การ Inheritance พร้อมยกตัวอย่างประกอบ (6 คะแนน)

1.4 จากข้อมูลทั้ง 10 จำนวนต่อไปนี้ จงวาดต้นไม้แบบ Binary Search Tree

13 11 1 3 5 7 9 4 2 12 (7 คะแนน)

1.5 จากการวาดต้นไม้ในข้อที่ 1.4 จงเขียนผลการเยี่ยมเยียน node ต่าง ๆ แบบ Pre-order และ In-order ตามลำดับ (7 คะแนน)

.....

.....

.....

.....

ข้อที่ 2 FILE PROCESSING *(45 คะแนน: 45 นาที)*

จงเขียนโปรแกรม เพื่ออ่านข้อมูลจากไฟล์ Book.txt แล้ว เขียนลงไปในไฟล์ NewBook.txt โดยเรียงลำดับตาม หมายเลขทางด้านซ้ายมือ และ ลบบรรทัดที่ซ้ำกันออกไป

Input file: Boox.txt

```
1       Computer Programming Techniques
4       Internet Engineering System
1       Computer Programming Techniques
2       Intelligent Security System
3       Killer Game Programming in Java
4       Internet Engineering System
3       Killer Game Programming in Java
```

Output file: NewBook.txt

```
1       Computer Programming Techniques
2       Intelligent Security System
3       Killer Game Programming in Java
4       Internet Engineering System
```

Note: สามารถเลือกใช้ Standard Template Library มาใช้งานได้ตามความเหมาะสม

ข้อที่ 3 DATA STRUCTURE WITH POINTERS *(25 คะแนน: 25 นาที)*

กำหนดให้ class Major เป็น class ที่เก็บข้อมูลของ รหัสสาขาวิชา และ ชื่อสาขาวิชา ซึ่งมีการสร้าง List Node ที่ใช้เก็บข้อมูลของ Major แล้วนำ List Node มาสร้างเป็น Link List ซึ่งจะใช้สำหรับค้นหาสาขาวิชา โดยค้นหาจาก ชื่อสาขาวิชา จงเขียนส่วนของโปรแกรมที่ขาดหายไป (2 จุด) ให้สามารถเรียกใช้งานจากฟังก์ชัน Main ได้อย่าง ถูกต้อง

```
//Major.h
#include <iostream>
#define MAX 50
class Major {
        public:
                Major( int , char * );
                int getId() const;
                char *getName();
        private:
                int id;
                char name[MAX];
};
Major::Major(int idCopy , char *nameCopy) {
     id = idCopy;
    strcpy(name,nameCopy);
}

int Major::getId() const{
        return id;
}
char *Major::getName(){
        return name;
}
```

```
//ListNode.h

#include <iostream>
#include "Major.h"

class ListNode {
        friend class List;
        public:
                ListNode(const Major &);
                Major getData() const;
        private:
                Major data;
                ListNode *nextPtr;
};


ListNode::ListNode(const Major &obj)
   :data(obj),nextPtr(0)
{
   //empty body
}
```

```
Major ListNode::getData() const
{
        return data;
}


class List {

        public:
                List();
                ~List();
                void insertAtFront( const Major &);
                bool isEmpty() const;
                void searchMajorByName(char *) const;
        private:
                ListNode *firstPtr;
                ListNode *lastPtr;
                ListNode *getNewNode(const Major &);

};

List::List(): firstPtr(0), lastPtr(0)
{
}

List::~List()
{
        if( !isEmpty() ) {
                ListNode *curPtr = firstPtr;
                ListNode *tempPtr;
                while (curPtr != 0) {
                        tempPtr = curPtr;
                        curPtr = curPtr->nextPtr;
                        delete tempPtr;
                }
        }
}

void List::insertAtFront( const Major &value)
{
   .....(1)..... (10 คะแนน)









}
```

```
bool List::isEmpty() const
{
   return firstPtr == 0;
}

ListNode *List::getNewNode(const Major &value)
{
   return new ListNode(value);
}

void List::searchMajorByName(char *major) const
{
   .....(2)..... (15 คะแนน)









}
```

```
//Main.cpp
#include <new>

#include "ListNode.h"

int main()
{
        int id[4]={1,4,3,2};
        char *major[4]={"network","infomation","csd","control"};
        char *searchMajor;
        List listA;
        for (int i=0;i<4;i++)
             listA.insertAtFront(Major(id[i],major[i]) );
        std::cout << "Enter major name: ";
        searchMajor = new char(50);
        std::cin >> searchMajor;
        listA.searchMajorByName(searchMajor);
        delete searchMajor;
}
```

Output

```
$ ./a.out
Enter major name: csd
Found Id:3 Major: csd
```

ข้อที่ 4 POLYMORPHISM & STL *(80 คะแนน: 80 นาที)*

องค์กรจำลองแห่งหนึ่ง มีพนักงาน (Employee) สองประเภท คือ `Hourly` และ `Volunteer` โดยข้อมูลที่สนใจของพนักงานแต่ละคน คือ การคำนวนเงินค่าตอบแทน

- Hourly เป็นพนักงานรายชั่วโมง โดยจะคำนวนเงินค่าตอบแทนตามจำนวนชั่วโมงที่ทำงานจริง โดยพนักงานแต่ละคนจะมีค่าตอบแทนต่อชั่วโมงไม่เท่ากัน ทั้งนี้หากทำงานเกินสัปดาห์ละ 40 ชั่วโมง จะได้รับค่าตอบแทนต่อชั่วโมงเป็น 1.5 เท่า

- Volunteer เป็นอาสาสมัคร ซึ่งจะไม่ได้รับค่าตอบแทนแต่อย่างใด

พนักงานทั้งสองประเภทจะนับจำนวนชั่วโมงทำงานผ่านฟังก์ชัน

```
void work(double hours);
```

โค้ดที่กำหนดให้ในตอนท้ายของข้อสอบชุดนี้ เป็นนิยามของ `class Employee` ซึ่งมีฟังก์ชันสำคัญดังต่อไปนี้

- `double getHours()` คืนค่า จำนวนชั่วโมงทำงานพนักงาน
- `int getID()` คืนค่า หมายเลข ID ของพนักงาน ซึ่งจะถูกกำหนดโดยอัตโนมัติจากลำดับของการสร้างอ๊อบเจค Employee
- `char* getName()` คืนค่า ชื่อของพนักงาน
- `double getPaid()` คืนค่า เงินรวมที่พนักงานคนนั้นๆ จะได้รับ

จากข้อมูลที่กำหนดให้ จงตอบคำถามข้อ 4.1-4.6

4.1 เติมโค้ด ณ. `/**1**/` เพื่อนำ Employee แต่ละคนบรรจุไว้ใน Vector v (5 คะแนน)

4.2 เติมโค้ด ณ. `/**2**/` เพื่อหาค่าตอบแทนรวมของพนักงานทุกคน และแสดงผลทางหน้าจอ ดังข้างล่าง (15 คะแนน)

```
1 John gets $299
2 Henrry gets $136
3 Jenny gets $0
4 Grace gets $144
5 Phebe gets $0
Total = $579
```

4.3 เติมโค้ด ณ. `/**3**/` เพื่อกำหนดค่าเริ่มต้นต่างๆ ให้กับ member ของ class (10 คะแนน)

4.4 เติมโค้ด ณ. `/**4**/` เพื่อแสดงชื่อ และจำนวนเงินที่ได้รับ **อย่างเหมาะสม** (10 คะแนน)

4.5 เติมโค้ด ณ. `/**5**/` เพื่อเขียนนิยาม `class Hourly` ให้สอดคล้องกับข้อมูลทั้งหมดที่กำหนดให้ (20 คะแนน)

4.6 เติมโค้ด ณ. `/**6**/` เพื่อเขียนนิยาม `class Volunteer` ให้สอดคล้องกับข้อมูลทั้งหมดที่กำหนดให้ (20 คะแนน)

```
#include <iostream>
#include <vector>

#include "employee.h"
#include "hourly.h"
#include "volunteer.h"

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

int main()
{
  vector< Employee* > v;
  Employee* s[5];
  s[0] = new Hourly("John", 5);
  s[0]->work(48);
  s[0]->work(10);
  s[1] = new Hourly("Henrry", 4);
  s[1]->work(34);
  s[2] = new Volunteer("Jenny");
  s[2]->work(18);
  s[3] = new Hourly("Grace", 4);
  s[3]->work(36);
  s[4] = new Volunteer("Phebe");
  s[4]->work(20);

  /** 1 **/




  /** 2 **/









  for(int i = 0; i < 5; i++)
    delete s[i];

  return 0;
}
```

```
#ifndef EMPLOYEE_H
#define EMPLOYEE_H

#include <iostream>
#define NAME_LEN 100

using std::ostream;

class Employee{
  friend ostream &operator<<( ostream&, const Employee & );
  private:
    static int gen_id;
    int id;
    double hours;
    char name[NAME_LEN];
  public:
    Employee( const char * name);
    double getHours() const { return hours; }
    void work(double hours) { this->hours += hours; }
    int getID() const { return id; }
    const char* getName() const { return name; }
    virtual double getPaid() const = 0;
};

int Employee::gen_id = 1;

Employee::Employee (const char* name){
  /* 3 */









}

ostream &operator<<( ostream& out, const Employee & em){
  /* 4 */









}
#endif
```

```
#ifndef HOURLY_H
#define HOURLY_H

#include "employee.h"

#define BASE_HOURS 40
#define BONUS_RATE 0.5
/** 5 **/









#endif
```

```
#ifndef VOLUNTEER_H
#define VOLUNTEER_H

#include "employee.h"

/** 6 **/









#endif
```