序号	

2020 学年度秋季学期试卷(A 卷)

课程名: 面向对象程序设计 课程号: 08305121 学分: 5

得分	

一、判断题（每小题 2 分，共 20 分）

1. C++程序中，任何类都有默认的构造函数。 ( )
2. C++程序中，非静态成员函数中的隐含形参 **this** 是一个指针变量，可以执行 **this**++ 运算。 ( )
3. C++程序中，类的非静态**常量成员函数**的隐含 **this** 指针为常量指针常量。即为 **const** *类名* * **const this**。 ( )
4. C++程序中，某函数的形参为**值传递**对象，则调用该函数时，程序会自动调用拷贝构造函数用实参对象拷贝构造形参对象；函数返回时，自动调用析构函数析构形参对象。 ( )
5. C++程序中，某函数的形参为引用型形参，则该引用的生命周期为函数调用时存在，函数返回后不存在。在引用的生命周期内，该引用绑定的实参变量（对象）不可更改。 ( )
6. C++程序中，可以用基类的对象初始化派生类的引用声明。 ( )
7. C++程序中，基类的所有成员(包括私有成员)在派生类中都可直接访问。 ( )
8. C++程序中，可以创建抽象类的对象。 ( )
9. C++程序中，重载运算符函数时，不能改变运算符的优先级。 ( )
10. C++程序中，重载运算符函数时，不能改变运算符的操作数个数。 ( )

得分	

二、填空题（每空 2 分，共 20 分）

请根据运行结果，完成程序。

```
#include <iostream>
#include <cstring>
using namespace ①_____;
template <typename TYPE> void Swap(TYPE &a, TYPE &b)
{                                                    // 函数模板
   TYPE temp;
   temp = a;
   a = b;
   b = temp;
}
void Swap(char *str1, char *str2)          // 重载函数模板
{
   char *temp = new char[strlen(②_____)+1];
   strcpy(temp, str1);
   strcpy(str2, temp);
   strcpy(str1, str2);
   ③_____ [] temp;
}
template <④_____ TYPE>
void Show(const TYPE &a, const TYPE &b)
{
   cout << a << ", " << b << endl;
}
int main()
{
   char c1='A', c2='B';
   int a=3, b=5;
   double x=3.3, y=5.5;
   ⑤_____ str1[80] = "Tom", str2[80] = "Jerry";
```

```
    const char *p1 = "Hello", *p2 = "你好";
    Swap(⑥_____, c2);       Show(c1, c2);
    Swap(⑦_____, b);     Show(a, b);
    Swap(⑧_____, y);     Show(x, y);
    Swap(⑨_____, str2); Show(str1, str2);
    Swap(⑩_____, p2);       Show(p1, p2);
    return 0;
}
```

运行结果(2)

```
B, A
5, 3
5.5, 3.3
Jerry, Tom
你好, Hello
```

得分 ____

三、阅读程序写出运行结果及简答题（共 30 分）

3.1（5 分）

```
#include <iostream>
using namespace std;
//提示:127 = 1×10^2 + 2×10^1 + 7×10^0 = 7×16^1 + 15×16^0 = 3×36^1 + 19×36^0
//127 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 1×3^4 + 1×3^3 + 2×3^2 + 0×3^1 + 1×3^0

char *Trans(unsigned int n, int base, char *str)
{
    char temp, BASE[] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    int i=0, j=0;
    if(n==0)
    {
        str[0] = '0';
        str[1] = '\0';
        return str;
    }
    while(n!=0)
    {
        str[i] = BASE[n%base];
        n /= base;
        i++;
    }
    str[i] = '\0';                    // 生成了 C-字符串 str
    for(j=0,i--; j<i; j++,i--)    // 将 C-字符串 str 的各字符（内容）倒置
    {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
    }
    return str;
}
int main()
{
    char str[80];
    unsigned int n = 127;
    cout << n << " = (" << Trans(n,  2, str) << ")_2" << endl;
    cout << n << " = (" << Trans(n,  3, str) << ")_3" << endl;
    cout << n << " = (" << Trans(n, 10, str) << ")_10" << endl;
    cout << n << " = (" << Trans(n, 16, str) << ")_16" << endl;
    cout << n << " = (" << Trans(n, 36, str) << ")_36" << endl;
    return 0;
}
```

运行结果(3.1)

```
127 = (_____)_2
127 = (_____)_3
127 = (_____)_10
127 = (_____)_16
127 = (_____)_36
```

3.2 （5 分）

```
#include <iostream>
using namespace std;
class Point
{
public:
    Point(int x=0, int y=0) : a(x), b(y)
    {
        cout << "构造 " << *this << endl;
    }
    virtual ~Point()
    {
        cout << *this << (a==b? "":"不") << "在直线 y = x 上" << endl;
    }
    friend ostream & operator<<(ostream &out, const Point &p)
    {
        out << "(" << p.a << ", " << p.b << ")";
        return out;
    }
```

运行结果(3.2)

```
_____
_____
_____
_____
_____
```

```
private:
   int a, b;
};
int main()
{
   Point x(2, 3), y(3, 3);
   cout << "返回操作系统" << endl;
   return 0;
}
```

3.3 **（5 分）**如下程序能演奏“致爱丽丝”的前 9 个音符。程序采用多文件结构，共有一个头文件、两个源程序文件。程序中使用了标准 MIDI 函数（打开/关闭 MIDI 设备、选择音色即选择乐器、开音/关音等），暂时不必深究这些函数。

文件之一（头文件）

```
// MIDI.h
// 本程序需要连接多媒体库 libwinmm.a。设置方法：
// Project/Settings.../Link/Libraries 项的值为 C:\MinGWStudio\MinGW\lib\libwinmm.a
#ifndef MIDI_H
#define MIDI_H
#include <windows.h>
#include <iostream>
#include <string>
using namespace std;
const int N = 100;
struct Music
{
   int deltaTime, note, volume;          // 时长（拍数）、音符、音量
};
class MIDI                // 乐器数字化接口(Musical Instrument Digital Interface)
{
public:
   MIDI(const string &Name="", int n=0, const Music *data=NULL);
                                          // 构造函数
   MIDI(const MIDI &m);                   // 拷贝构造函数
   virtual ~MIDI();                       // 析构函数（虚函数）
   virtual void Play() const;             // 演奏乐曲（虚函数）
protected:
   static DWORD MidiOutMessage(int iStatus, int iChannel,
                               int iFlip, int iVolume);
// 给 MIDI 设备发送消息,用于设置音色、发音或停止发音。DWORD 为 unsigned long
   static HMIDIOUT hMIDI;     // 静态数据成员（MIDI 设备句柄）
   static int NumObjs;        // 创建对象的个数
   string name;               // 乐曲名
   int len;                   // 乐曲的音符数（不超过 N）
   Music x[N];                // 乐谱
};

class MIDI_Piano : public MIDI         // 钢琴类
{
public:
   MIDI_Piano(const string &Name="", int n=0, const Music *data=NULL);
   void Play() const;
protected:
   int timbre;                         // 音色（对应某种乐器）
};

class MIDI_Violin : public MIDI        // 小提琴类
{
public:
   MIDI_Violin(const string&Name="", int n=0, const Music*data=NULL);
   void Play() const;
protected:
   int timbre;                         // 音色（对应某种乐器）
};

class MIDI_ChurchOrgan : public MIDI  // 教堂管风琴类
{
```

```
public:
  MIDI_ChurchOrgan(const string&Name="",int n=0,const Music*data=NULL);
    void Play() const;
protected:
    int timbre;                              // 音色（对应某种乐器）
};
#endif
```

文件之二（源程序文件）

```
// MIDI.cpp
#include "MIDI.h"

HMIDIOUT MIDI::hMIDI;          // 静态数据成员定义（MIDI 设备句柄）
int MIDI::NumObjs = 0;         // 静态数据成员定义（对象的个数）

DWORD MIDI::MidiOutMessage(int iStatus,int iChannel,int iFlip,int iVolume)
{
    DWORD dwMessage = (iVolume<<16) | (iFlip<<8) | iStatus   | iChannel;
    return midiOutShortMsg(hMIDI, dwMessage);
}   // 本函数的功能是向 MIDI 设备发送消息，实现设置音色、设备发出声音等功能。无须深究。

MIDI::MIDI(const string &Name, int n, const Music *data)
                                          : name(Name), len(n)
{
    if(++NumObjs==1)             // 若是第一个对象，则打开 MIDI 设备
       midiOutOpen(&hMIDI, 0, 0, 0, CALLBACK_NULL);
    if(len<0) len = 0;
    if(data==NULL) return;
    for(int i=0; i<len; i++)
       x[i] = data[i];
}

MIDI::MIDI(const MIDI &m):name(m.name), len(m.len) // 拷贝构造函数
{
    ++NumObjs;
    for(int i=0; i<len; i++)
       x[i] = m.x[i];
}
```

```
MIDI::~MIDI()
{
    if(--NumObjs==0)                  // 若是最后一个对象，则关闭 MIDI 设备
       midiOutClose(hMIDI);
}
void MIDI::Play() const
{
    cout << name << endl;
    for(int i=0; i<len; i++)
    {
       cout << x[i].note << " ";
       MidiOutMessage(0x90, 0x00, x[i].note, x[i].volume);  //开音
       Sleep(x[i].deltaTime*60000/180);   // 使开音延时，每分钟 180 拍
       MidiOutMessage(0x80, 0x00, x[i].note, 127);          //关音
    }
    cout << endl;
}

MIDI_Piano::MIDI_Piano(const string &Name, int n,
          const Music *data) : MIDI(Name, n, data), timbre(0)
{
}
void MIDI_Piano::Play() const
{
    cout << "钢琴演奏\t";                          // 请注意此处无换行
    MidiOutMessage(0xC0, 0x00, timbre, 0);   // 设置音色（即乐器）
    MIDI::Play();
}

MIDI_Violin::MIDI_Violin(const string &Name, int n,
          const Music *data): MIDI(Name, n, data), timbre(40)
{
}
void MIDI_Violin::Play() const
{
    cout << "小提琴演奏\t";                        // 请注意此处无换行
```

```
   MidiOutMessage(0xC0, 0x00, timbre, 0);   // 设置音色（即乐器）
   MIDI::Play();
}

MIDI_ChurchOrgan::MIDI_ChurchOrgan(const string &Name, int n,
            const Music *data) : MIDI(Name, n, data), timbre(19)
{
}
void MIDI_ChurchOrgan::Play() const
{
   cout << "教堂管风琴演奏\t";                    // 请注意此处无换行
   MidiOutMessage(0xC0, 0x00, timbre, 0);   // 设置音色（即乐器）
   MIDI::Play();
}
```

文件之三（源程序文件）

```
// MIDI_TEST.cpp
#include "MIDI.h"
void Playing(const MIDI &m)
{
   m.Play();
}
int main()
{
   Music data[] = {{1,76,96}, {1,75,96}, {1,76,96}, {1,75,96},
         {1,76,96}, {1,71,96}, {1,74,96}, {1,72,96}, {3,69,96}};
   int n = sizeof(data)/sizeof(*data);

   MIDI_Piano p("致爱丽丝", n, data);
   MIDI_Violin v("致爱丽丝", n, data);
   MIDI_ChurchOrgan c("致爱丽丝", n, data);

   Playing(v);
   Playing(c);
   Playing(p);
   return 0;
}
```

运行结果(3.3) 部分结果已给出

```
76 75 76 75 76 71 74 72 69
```

3.4 对于 3.3 题中的代码，简要回答如下问题**（15 分，每小题 3 分）**

(1) 能否创建基类（MIDI 类）的对象？为什么？

(2) MIDI 类中为什么必须定义拷贝构造函数（尽管本程序尚未测试拷贝构造函数）？

(3) MIDI 类中的静态数据成员 NumObjs 的作用是什么？

(4) MIDI 类中，若将 Play 函数设计成非虚函数，则程序运行会产生什么效果？

(5) MIDI 类中，若将 Play 函数设计成非常量成员函数，程序的其他部分不修改，则程序能否通过编译？若不能通过编译，请指出哪个文件的哪条语句出错，为什么？

得分

四、编写程序（共 30 分）

有如下尚未完成的代码，其中已完成构成单向链表类模板所需的结点类模板(**template** <**typename** T> **class** Node;)的设计；单向链表类模板(**template** <**typename** T> **class** LinkList;)中数据成员、成员函数原型已设计。请在类体外完成①~⑧函数的定义（①~⑥每个函数 4 分，⑦~⑧每个函数 3 分）。

```
// LinkList.h
#ifndef LINKLIST_H
#define LINKLIST_H
#include <iostream>
using namespace std;

template <typename T> class LinkList;                    // 提前声明

template <typename T> class Node                         // 结点类模板
{
public:
   Node() : next(NULL) {}
   Node(const T &t) : data(t), next(NULL) {}
   Node(const Node<T> &node) : data(node.data), next(NULL) {}
   Node<T> & operator=(const Node<T> &node)
   {
      data = node.data;
      return *this;
   }
   friend class LinkList<T>;
private:
   T data;
   Node<T> *next;
};

template <typename T> class LinkList                     // 单向链表类模板
{
public:
   LinkList(int n=0, const T *x=NULL);                   // ① 构造函数
   LinkList(const LinkList<T> &link);                    // ② 拷贝构造函数
   virtual ~LinkList();                                  // ③ 析构函数
   LinkList<T> & operator=(const LinkList<T> &link);
                                                         // ④ 赋值运算符函数
   void FreeList();                                      // ⑤ 释放所有结点
   void Append(const T &t);                              // ⑥ 追加新的尾结点
   virtual void ShowList(ostream &out)const;             // ⑦ 输出
protected:
   Node<T> *head;
};                                                       // ⑧ 重载 operator<< 运算符
```