CPS 271 Midterm [100 points] Fall 2021 Tuesday, October 12, 2021

Name:____jinfang wei_____

You should upload as an attachment the .cpp files, the input, the output and a word document that has all the screen printouts.

Questions 1 – 13 – 2 pts each

Answer the next 5 questions based on the following program:

```
#include <iostream>
using namespace std;

class Distance {
    static int feet;
      int inches;
public:
    Distance(int f=0, int i=0): inches(i){ feet = f}
    static int getFeet() { return feet; }
    void display (){ cout << feet << " - " << inches << endl;}
};
int Distance::feet = 0;
```

- Write a statement to declare dist1 to be a Distance object and initializes its member variable feet to 20 and inches to 10, assuming the code has no compilation errors.

Distance dist1(20, 10);

- Write a statement to declare dist2 to be an instance of Distance class.

Distance dist2;

- Write statements to display both dist1 and dist2.

dist1.display();
dist2.display();

- Write a statement to define a pointer to a Distance.

Distance *dist = new Distance;

- Which of the following is valid (assume dist1 is declared as an instance of Distance class):
  - `dist1.getFeet;`
  - `Distance.getFeet();`
  - `dist1::getFeet();`

- **Distance::getFeet();**
- none of the above.

Answer the next 4 questions based on the following program:

```
 1: class Mystr
 2: {
 3: public:
 4:     Mystr(const char * s);
 5:     ~Mystr();
 6:     Mystr(const Mystr & m);
 7:     Mystr();
 8:     Mystr(int len, char* a);
 9:     int length() const;
10:     char & operator [] (int n);
11:     operator const char *() const;
12:     Mystr operator = (const Mystr &operand);
13:     Mystr operator + (const Mystr &operand);
14:     friend Mystr operator + (const char * str,const Mystr
&op);
15:     void  operator += (const Mystr &operand);
16:     bool operator == (const Mystr &m) const;
17:     bool operator != (const Mystr &m) const;
18: };
```

```
//Assume the following code in main:
Mystr mystra("xyz"), mystrb("XYZ");
int y = 0;
char ch = ' ';
```

6. What line number will be executed if we enter the code: `if (mystra == mystrb){ y = 2;}`
Line16

7. What line number will be executed if we enter the code: `mystra = mystrb;`
Line12

8. What line number will be executed if we enter the code: `mystra[2] = 'K';`
Line10

9. What line number will be executed if we enter the code: `ch = mystra[1];`
Line11
I have question on 8 and 9,not sure the answer.

10. What is the output of the following program?

```
int x[] = {1, 2, 3, 4, 5, 6};
int *p = &x[3];
for (int i = 1; i < 3; i++)
   cout << *p;
```

44;
if cout<<*p<<endl; would be better to understand,
4
4

11. What is the output of the function `doit()` if it was called?
PSAD

```
class Person {
 public:
   Person() {
      cout << 'P';
   }
   ~Person() {
      cout << 'D';
   }
   void abcd(double x) {
      cout << 'A';
   }
   void abc (char c) {
      cout << 'E';
   }
};

class Student : public Person {
  public:
   void abc (char c) {
      cout << 'S';
   }
};

void doit () {
   Student s;
   s.abc('z');
   s.abcd(1.2);
}
```

12. What is the output of the following code?

a fat fox

```
string str = "the fat fox jumped over a fat hedge";
string str2 ="a ";
str2 += str.substr(4,7);
cout << str2 << endl;
```

13. Which of the following four classes properly initialize their member variable x to 1? Assume that the default constructor will be used for creating your object:

```
class myclassa{
   int x=1;
};

class myclassb{
    int x;
 public:
    myclassb(){
       x = 1;
    }
};

class myclassc{
    int x;
 public:
    myclassc(): x = 1;
    {}
};


class myclassd{
   int x;
   x = 1;
};
```

- myclassa
- **myclassb**
- myclassc
- myclassd
- none of the above

Questions 14 – 23 – 4 pts each

Write a definition for each item **AND** give a short example of how it would be used:

14. const member function

The const member functions are the functions which are declared as constant in the program. The object called by these functions cannot be modified. It is recommended to use const keyword so that accidental changes to object are avoided.
datatype function_name const();

15. default copy constructor

A special constructor for creating a new object as a copy of an existing object.
```
Point::Point(const  Point&p){
x=p.x;
y=p.y;
}
```
default copy constructor means no variable in paramter, like
```
Point::Point(){ };
```

16. pass by reference

Pass-by-reference means to pass the reference of an argument in the calling function to the corresponding formal parameter of the called function. The called function can modify the value of the argument by using its reference passed in.
```
void f(const int& x) {
  int& y = const_cast<int&>(x);
  ++y;
}
```

17. operator overloading

When you create new functionality for an existing operator. Like for the Fraction we defined how to + two fractions. Those are +-*/ operator.
```
Fraction operator+(Fraction f);
Fraction operator-(Fraction f);
Fraction operator*(Fraction f);
Fraction operator/(Fraction f);
```

18. HasA relationship

The use of instance variables that are references to other objects.
```
Person::Person(std::string namea, Mydate birthdate, std::string phoneNoo,
```
In this case person has mydate . Just compistion of part person class.

19. pass by value

A copy of the value is passed into the function so if it changes in the function, the new value does not come out of the function.
```
void foo(int y){cout<<y<<endl;}
int main(){
foo(5);
```

}

20. private class member

Only objects and functions within the class can access private members.
x, y called privte class member in this case;
class Example{
private:
int x;
int y;
}

21. IsA relationship

IS-A is a totally based on inheritance, which can be of two type of class inheritance or interface inheritance; like "A is a B type of thing" ; for example person is a employee.

```
class Employee : public Person
```

22. destructor

A destructor is a special member function that is called when the lifetime of an object ends.

```
~Employee() {};
```

23. friend function

A friend function is a function that is not a member of a class but has access to the class's private and protected members.

```
friend std::ostream& operator<<(std::ostream& o, const Fraction& r);
friend std::istream& operator>>(std::istream& is, Fraction& r);
```

Questions 24 – 30 – 2 pts each

24. Consider the following class definitions:

```
class aClass{
 private:
  int x;
  int y;
 public:
    aClass ();
    aClass (int, int);
    void print() const;
    void set(int, int);
};
```

What is wrong with the following class definitions?

```
class bClass : public aClass {
 private:
  int z;

public:
   void print();
   void set(int, int, int);
};
```

This is what I did. It looks no special wrong thing, this is declaration, here is my answer for definitions. I confused this question. Also I add constructor in case crash.

```
bClass::bClass() {
	aClass::aClass();
	z = 0;

}

bClass::bClass(int x, int y, int z) :aClass(x, y) {
	this ->z = z;

}
void bClass::print() {
	aClass::print() const; //this is a error, but I don't know why

	cout << z << endl;
}
void bClass::set(int x, int y, int z) {
	aClass::aClass(x, y);
	this->z = z;
}
```

Consider the following class definition:

```
class aClass {
 private:
  int a;
  int b;
 public:
   aClass ();
   void print();
   void set(int, int);
};

class bClass: public aClass {
 public:
  int z;
public:
   void print();
   bClass();
```

```
};
```

and assume the following two objects were created in `main()`:

```
aClass y;
bClass x;
```

Mark the following statements as valid or invalid:

```
25. cout << x.set(15, 30);
```
invalid

```
26. cout << y.b << endl;
```
invalid

```
27. cout << x.z << endl;
```
invalid

```
28. x.set(15, 30);
```
invalid

29. What is wrong with the following code?

This print out *q and *p should put before the delete p

If it puts after delete memory q, p, nothing would print out and send wrong error.

```
int *p;
int *q;

p = new int; int*p=new int;
*p = 43;

q  = p;
*q  = 52;

cout << *p << *q << endl;

delete q;
```

30. The difference between a struct and a class is:

- A struct can't have functions.
- You can't use the new operator to create a struct. Classes can be dynamically created with the new operator.
- By default, everything in a class is private. By default, everything in a struct is public.
- A struct can't have a constructor.
- **None of the above.**

Fill in the blank (2 pts each):

31. A base class's **public** members can be accessed in the base-class definition, in derived-class definition and in friends of the base class and its derived classes.

32. When deriving a class from a base class with public inheritance, public members of the base class become **public** members of the derived class.

33. When overloading the << or >> operators for a class, those functions are not "members" of the class, but are **friend** functions of the class.

34. When deriving a class from a base class with protected inheritance, public members of the base class become **public** members of the derived class.

35. A static data member in a class definition requires that only **static** member functions can access that data member. **I am not sure this answer, I should learn static function.**

Question 36 – 10 pts

This is header file: for some reason, it could not read when I uploaded on blackborad. so I attached here.

```
#pragma once

#include <iostream>

#include <string>

using namespace std;


constexpr auto MAX_ARY = 500;


//const MAX_ARY = 500


class Entry {

private:

    int m_hour;

    int m_minute;

    string m_todo;


public:

    Entry();
```

```
    void setData(int hour, int minute, const char* td);
    void setData(Entry& e);

    //This routine prints a line that would look something like:
    // 12:30 Take a Walk
    void printEntry();

    int getHour();
    int getMin();
    string getToDo();
};



class Schedule
{
private:
    // Array of Entry
    Entry* m_entryArr;
    int m_maxEntries;

    int m_actualEntryCount;

public:

    Schedule(int maxEntries);


    Schedule(Schedule& s);
    ~Schedule();
```

```
    bool addEntry(int hour, int minute, const char* todo);

    void printSchedule();

};
```

36. You are given a main.cpp and a Schedule.h file. You are to write a Schedule.cpp file that implements the missing pieces from Schedule.h.

```
// ************************************
// *************  main.cpp ************
// ************************************
#include "Schedule.h"
void main()
{
 Schedule s1(10);
 s1.addEntry(1,20,"Feed Cat");
 s1.addEntry(2,40,"Feed Dog");
 s1.addEntry(2,50,"Walk Dog");
 s1.addEntry(4,0, "Fix Dinner");
 s1.addEntry(5,15,"Eat Dinner");
 s1.printSchedule();
 Schedule s2(s1); // Note this uses the copy constructor
 cout << endl << "Output from s2 " << endl;
 s2.printSchedule();
}


// ********************************************
// *************   Schedule.h  ****************
// ********************************************
#include <iostream>
#include <string>
using namespace std;

#define MAX_ARY 500

class Entry{
private:
```

```
    int m_hour;
    int m_minute;
    string m_todo;

public:
    Entry()
    {
        m_hour = -1;
        m_minute = -1;
        m_todo = "N/A";
    }

    void setData(int hour, int minute, const char *td);
    void setData(Entry & e);

    //This routine prints a line that would look something
like:
    // 12:30 Take a Walk
    void printEntry();
};

class Schedule
{
private:
    // Array of Entry
    Entry m_entryArr[MAX_ARY];

    // User defined size of the Array
    int m_maxEntries;

    // Number of entries filled in with setData (starts out
= 0)
    int m_actualEntryCount;

public:
// This Constructor will initialize 2 ints and define the
m_entryArr. Make sure MAX_ARY is not exceeded
    Schedule(int maxEntries);

// Copy Constructor - it is done for you
 Schedule(Schedule & s) {
       m_actualEntryCount = s.m_actualEntryCount;
       m_maxEntries = s.m_maxEntries;
       for (int i=0; i < m_actualEntryCount; i++)
       {
             m_entryArr[i].setData(s.m_entryArr[i]);
       }
```

```
 }
  ~Schedule();
// addEntry returns true if it succeeds, false if we are
out of space
// Note that addEntry will merely call setData to fill in
the next entry
// in the array and increment the m_actualEntryCount
 bool addEntry(int hour, int minute, const char *todo);

// This routine prints out all entries (i.e. as specified
by m_actualEntryCount)
 void printSchedule();
};
```