LAB 5: INTRODUCTION TO CLASSES (ANSWERS/ CODES)

Program 5.1 – Program 5.5: No answers/ codes

Program 5.6

Code:

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

class Date
{
  private:
        int day;
        int month;
        int year;
  public:
        //(i)Default constructor prototype
        Date();

        //(ii)Constructor with 3 parameters prototype
        Date(int, int, int);

        int getDay ()
        { return day;}
        int getMonth ()
        { return month;}
        int getYear ()
        { return year;}
 };

//(i) Default constructor definition
Date::Date()
{
    day = 1;
    month = 1;
    year = 90; // default to 1990
}

//(ii) Constructor with 3 parameters definition
Date::Date(int inputDay, int inputMonth, int inputYear)
{
    day = inputDay;
    month = inputMonth;
    year = inputYear;
}


int main()
{
    //(iii) Creation of Date object called birthDate
    Date birthDate(13, 8, 92);

    /* (iv) Creation of Date object pointer called
    datePtr */
    Date* datePtr;
```

```
50.
51.     // points to birthDate object
52.     datePtr = &birthDate;
53.
54.     // prints out birth day, month, and year
55.     cout << "My birth day is: " << datePtr->getDay() << endl;
56.     cout << "My birth month is: " << datePtr->getMonth() << endl;
57.     cout << "My birth year is: " << datePtr->getYear() << endl;
58.
59.     return 0;
60. }
```

Output:

My birth day is: 13

My birth month is: 8

My birth year is: 92

Program 5.7

Output:

5.5 94.985

Explanation:

Standalone function printCircle() does not have to be friend of class Circle to access the getRadius() and getArea() functions of class Circle because they are public.

Program 5.8

Output:

5.5 94.985

Program 5.9

Output:

5.5 94.985

Explanation:

The function printCircle() takes a Circle pointer as input. Since it is a pointer, it needs to use the -> operator to access the member function of the object that it is pointing to.

Program 5.10

Output:

0

Explanation:

The for loop loops for 10 times, calling increment() function each time. The increment() function is supposed to increase the pay variable of object by 1 each time. The pay variable of object is initialized to 0 by the default constructor, after looping 10 times; the pay variable of the object should be 10.

However, this does not happen. This is because the function needs to accept an address of the object to change the value of the object. This can be done by putting Bonus &p as the parameter.

Program 5.11

Output:

7.5

Explanation:

c1 is an array of Cone objects. Cone objects in the array can be initialized by number e.g. 4.5 or Cone (number) e.g. Cone(5.5). Next, for loop loops 3 times and call getRadius() of each Cone object of c1 array, i.e. c1[0], c1[1], and c1[2]. This will return the radius of each object.

Cone c2 is initialized by default constructor to have radius of 3.5. We call longerRadius() on c2, and passing c1 array. The function loops through the array elements and compare each object's radius value with the next object's value, if it is bigger, it will be stored in the longest variable. This will find the longest radius among the objects in the array. This is then printed.

Program 5.12

Output:

Enter the faculty of your major:

(in numeric code):1

Enter the faculty of your minor:

month (in numeric code):9

Your major is in the Faculty of 1 Computing.

Your minor is in the Faculty of 9 Science.

Your records are in order.

Explanation:

A Faculty object stores the faculty numeric code in fac, and it has display() function to display the faculty name by matching code using switch statements. Two objects of class Faculty is used to store and display the major and minor of student. The major and minor object is checked if their fac variable value is same, if it is, error is printed.

Program 5.13

Explanation:

The question is, which of the statements are valid? i. is not valid because it tries to access a private member variable, price. v. and vi. are not valid because they try to access a private member function, getdiscount(). vii. is not valid because we cannot compare two objects, unless we overload the operator <. ii., iii., iv., and ix. are valid. ix. is valid because it is a memberwise assignment, eventhough the variables are all private, it is still valid.

Program 5.14 – answers at Program 5.15, 5.16

Program 5.15

Output:

The object has just been created

Weight: 150

Height: 120

The object has just been created

Weight: 175

Height: 74

The object has just been created

Weight: 0

Height: 0

The area of b1 is 0.0104167

The area of b2 is nan

The area of b3 is 0.0319576

The object is being destroyed

Weight: 175

Height: 74

The object is being destroyed

Weight: 150

Height: 120

Explanation:

b1 object is created and BMI constructor accepts two double, then initialized weight and height to 150, and 120 respectively. Two pointers, b2, and b3 is created, b3 points to an object with weight and height set to 175, and 74 respectively. b2 points to object that is initialized by default constructor to have weight and height set to both 0. Then, the getBMI() function of b1 is called by

using dot operator, getBMI() function of object pointed to by b3 is called using -> operator, same to b2. This prints the BMI of all three. Then, delete b3; statement is executed, this will cause the destructor of object pointed by b3 to execute, printing “The object is being destroyed” followed by weight and height. When return 0; statement is executed, destructor of remaining object, b1 is executed.

Program 5.16

Output:

The object has just been created

Weight: 180.2

Height: 132

The object has just been created

Weight: 155.4

Height: 69.6

The object has just been created

Weight: 139

Height: 100

The area of b1 is 0.0103421

The area of b2 is 0.0320799

The area of b3 is 0.0139

The object is being destroyed

Weight: 139

Height: 100

The object is being destroyed

Weight: 155.4

Height: 69.6

The object is being destroyed

Weight: 180.2

Height: 132

Explanation:

b is an array of BMI objects. b is initialized with three BMI objects, which are initialized by BMI constructor that sets the weight and height. Then, the getBMI() function of each object is called. This will print out the BMI of each object. When the return 0; statement is encountered, destructor of all objects will be executed, causing three “The object is being destroyed” followed by weight and height to be printed.

Program 5.17

Output:

You are in function main

You are now in the constructor of object Lecturer

After creating job1

Before calling to func()

You are now in func()

You are now in the constructor of object Technician

After creating job2

You are now in the destructor of object Technician

After calling to func()

You are now in the destructor of object Lecturer

Explanation:

When the line JobDesc job2("Lecturer"); is executed, a job2 object of class JobDesc is created, a C-String is passed to the constructor. The C-string is copied into jobTitle character array by strcpy() function.

When func() is called, it creates an object job2 of class JobDesc is created with “Technician” C-String passed to constructor. It is similar to just now but after the function is returned, the object created in the function scope (job2 with “Technician” C-String) is destroyed, hence, destructor is called.

After return 0; is executed, destructor of object job2 with “Lecturer” C-String is called.

Program 5.18

Output:

Invisible Box

Length = 0 Width = 0 Heigth = 0

Mystery Box:

Winning Box 7

Length = 7 Width = 7 Heigth = 7

Winning Box 0

Length = 0 Width = 0 Heigth = 0

Smashed Box

Smashed Box

Smashed Box

Explanation:

platinumBox is an object of class Box, when it is created, a default constructor is called as there is no input, it sets the length, width, and height to 0 and prints “Invisible Box”. print() function of platinumBox is called, printing the length, width, and height.

Next, goldBox object of class Box is created, constructor is called by accepting the 7 (int) and sets the length, width, and height to 7 and prints “Winning Box” followed by the int value.

silverBox object is similarly created.

When return 0; is encountered, all the objects is destroyed and destructors are called, printing 3 “Smashed Box”.

Program 5.19, 5.20 – answers at Program 5.21

Program 5.21

Output:

Enter today’s date:

Month (in number):4

day (in number):17

Today is: 17 April.

Malaysia’s Independence Day.31 August.

Selamat Datang! Welcome to Malaysia.

Explanation:

Two objects dateToday and independenceDay of class DayOfYear is created. The first is created using default constructor where the month and day is set to 1. The second is created using constructor that accepts two int input, 8, 31, and set the month and day to that.

As the name suggests, dateToday object will be used to store today’s date input by user and independenceDay is Merdeka Day which is 31^{st} of 8^{th} month.

The enter() function of dateToday is called to asks user to input month and day, check_date() function is used to check if the month and day is valid. Then, display() function of dateToday is called to display the day and month name using switch statements. day and month name of independenDay is also displayed.

Lastly, same() function is a standalone function with access to (friend of) DayOfYear class. It accepts dateToday and independenceDay object and compare their month and day variables.

Program 5.22 – answers at Program 5.23

Program 5.23

Code:

```
// class member function implementation

// Constructor 1 (default constructor) sets the name to
// NULL and the maxSubject to DEFAULT_SIZE
Student::Student()
{
   name = '\0';
   maxSubject = DEFAULT_SIZE;
   subjects = new SubjectCode[maxSubject];
   subjectCount = 0;
}

// Constructor 2 sets the name with n
//    and the maxSubject to DEFAULT_SIZE
Student::Student(const char n[])
{
    name = new char[20];
    strcpy(name,n);
    maxSubject = DEFAULT_SIZE;
   subjects = new SubjectCode[maxSubject];
    subjectCount = 0;
}

// Constructor 3 sets the maxSubject to max
//    and the name to NULL
Student::Student(int max)
{
    name = '\0';
    maxSubject = max;
   subjects = new SubjectCode[maxSubject];
   subjectCount = 0;
}

// Constructor 4 set the name to NULL and insert subject
//   codes to array 'subjects' using data 'codes'. //
//   The maxSubject is set to count.
Student::Student(const SubjectCode codes[], int count)
{
    name = '\0';
    maxSubject = count;
   subjects = new SubjectCode[maxSubject];
    for(int i=0; i<count; i++)
    {
        strcpy(subjects[i],codes[i]);
    }
   subjectCount = count;
}

// Destructor free the name and subjects. It also
//   prints a message 'Object â€¦. is destroyed
Student::~Student()
{
    delete name;
    delete subjects;
    cout << "Object is destroyed" << endl;
}
```

```
57.
58. // Accessor Functions
59. //returns the name
60. const char * Student::getName() const
61. {
62.     return name;
63. }
64.
65. // returns the subject code of index 'index'
66. const char * Student::getSubject(int index) const
67. {
68.     return subjects[index];
69. }
70.
71. // Print the object information containing
72. // Name, Maximum subject could be taken,
73. // Number of subjects taken and
74. //   List of subjects taken:
75. void Student::print() const
76. {
77.     cout << "Name: " << name << endl;
78.     cout << "Maximum subject could be taken: " << maxSubject << endl;
79.     cout << "Number of subjects taken: " << subjectCount << endl;
80.     cout << "List of subjects taken: ";
81.     for(int i=0; i<subjectCount; i++)
82.     {
83.         cout << subjects[i] << " ";
84.     }
85.     cout << endl;
86. }
87.
88. // Mutator Functions
89. // Sets the student name
90. void Student::setName(const char newName[])
91. {
92.     name = new char[20];
93.     strcpy(name,newName);
94. }
95.
96. // Insert a subject code into the array
97. //  'subjects'.  This function must firstly check
98. //   either the array the array is already full.
99. //   The array is full if subjectCount reaches
100.    //   the maxSubject
101.    void Student::insertSubject(const SubjectCode code)
102.    {
103.        if(subjectCount >= maxSubject)
104.        {
105.            return;
106.        }
107.
108.        strcpy(subjects[subjectCount],code);
109.
110.        subjectCount += 1;
111.    }
112.
113.    // Replace the subject of index 'index'
114.    void Student::setSubject(int index, const SubjectCode code)
115.    {
116.        strcpy(subjects[index],code);
117.    }
```

Output:

Name: Afiq

Maximum subject could be taken: 5

Number of subjects taken: 3

List of subjects taken: SCSJ1223 SCSJ1243 SCSJ1253

The first subject taken by Afiq is SCSJ1223

Name: Ahmad Hussein

Maximum subject could be taken: 5

Number of subjects taken: 2

List of subjects taken: SCSJ2023 SCSJ2343

Ahmad Hussein has changed his/her second subject to SCSJ3633

Name: Husna

Maximum subject could be taken: 3

Number of subjects taken: 3

List of subjects taken: SCSJ1223 SCSJ1243 SCSJ1253

Name: Abu Bakar

Maximum subject could be taken: 4

Number of subjects taken: 4

List of subjects taken: SCSJ1123 SCSJ3633 SCSJ3023 SCSJ2623

Object is destroyed

Object is destroyed

Object is destroyed

Object is destroyed