MINIA UNIVERSITY

FACULTY OF COMPUTER & INFORMATION SCIENCES

Object-Oriented Programming

Assignment #1

Define a class***fraction*** that has two data members:***num*** and***denom***, (both of type

**int**), which represent the fraction’s numerator and denominator, respectively. It has

the member function***getFract()*** that reads the***num*** and***denom*** of a***fraction*** object

from the keyboard in fractional form (i.e. num/denom), the member function

***showFract()*** that displays a***fraction*** object in fractional form, and the member

functions:***addFract()*** ,***subFract()***,***mulFract()***,***divFract()***, for the performing

addition, subtraction, multiplication and division operations on any two objects of

class***fraction***, respectively.

Write a program that declares two objects of class***fraction***, reads their data from the

keyboard, then perform the four operations on them, and displays the result in each

case.

**Notes:**

(1) The addition of the two fractions a/b and c/d can be performed by the

formula:

*a*

*b*



*c*

*d*



*a* \**d* *b* \**c*

*b* \**d*

(2) The subtraction of the two fractions a/b and c/d can be performed by the

formula:

*a*

*b*



*c*

*d*



*a* \**d* *b* \**c*

*b* \**d*

(3) The multiplication of the two fractions a/b and c/d can be performed by the

formula:

*a*

*b*



*c*

*d*



*a* \**c*

*b* \**d*

(4) The division of the two fractions a/b and c/d can be performed by the

formula:

*a*

*b*



*c*

*d*



*a* \**d*

*b* \**c*

MINIA UNIVERSITY

FACULTY OF COMPUTER & INFORMATION SCIENCES

Object-Oriented Programming

Assignment #2

(a) Define a class***counter***, which has three data members:***count*** (of type int) that

represents the value of the counter,***overflow*** (of type bool) that is set to*true*

if an overflow occurs, and***maxcnt*** (of type int) that represents the maximum

value of the counter. It has a constructor***counter(int max\_count)***, that

creates a counter with a given maximum count and an initial value 0, and sets

*overflow* to*false*, and it has 4 member functions:***increment()***, which

increases the current count by 1; if the current count equals the maximum

count, the*count* is set to zero, and*overflow* to true;***reset()***, which sets the

counter to zero and overflow to false;***get\_count()***, which returns the current

count, and***check\_for\_overflow()***, which returns true if an overflow has

occurred and false if it has not.

(b) Using class***counter***, write a main program that reads a sentence, and counts the

frequency of each vowel (a, e, i, o, u) in it, then displays these frequencies. The

program should use a separate counter for each vowel, and it should display an

error message if any of the counters overflows.

MINIA UNIVERSITY

FACULTY OF COMPUTER & INFORMATION SCIENCES

Object-Oriented Programming

Assignment #3

**Part A:**

A set is a collection of values; the order of the values in the set is irrelevant, and

duplicate values are not allowed. Define a class**set**, whose objects are sets of integers.

Class**set** has two data members:***elem*** (array of integers) that holds the elements of

the set, and***size*** (of type int) that holds the number of elements in the set. Class**set**

has a constructor***set()***, which creates an empty set, a member function***is\_element()***,

which returns true if a given value belongs to a set object, a member function***empty()***,

which returns true if a set object is empty, a member function***show\_set()***, which

displays the elements of a set object. Also, class**set** has the following operators:

 **Operator +** yields the*union* of two sets, which contains those elements that

belong to either or both sets.

 **Operator –** yields the*difference* of two sets, which contains those elements

that belong to the first set but not to the second.

 **Operator \*** yields the*intersection* of two sets, which contains those elements

that belong to both sets.

 **Operator =** copies a given set into another one.

 **Operator +=** adds a given value to the set if it does not already belong to it.

 **Operator -=** removes a given value if it belongs to the set.

 **Operator <** compares two sets and returns true if the first set is a subset of the

second, i.e. if every element of the first set also belongs to the second.

 **Operator >** compares two sets and returns true if the first set includes the

second, i.e. if every element of the second set also belongs to the first.

 **Operator ==** compares two sets and returns true if they have the same

elements.

**Part B:**

Using the class***set***, write a main program that performs the following tasks:

1. Create two objects, set1 and set2, of the class set.

2. Read the elements of the two sets: set1 and set2.

3. Find the union of the two sets, store it in another set object and display it.

4. Find the differences (set1 – set2) and (set2 – set1), store them in two set

objects and display them.

5. Compare the two sets and display one the following messages accordingly:

 The two sets are identical.

 The first set is a subset of the second set.

 The first set includes the second set.

 The two sets are not identical.

6. Find the intersection of the two sets, and store it in another set object. If the

intersection is not empty, display it, and remove the common elements from

both sets, then display the resultant two sets.

Run your program with different data to cover all the possibilities to make sure that

all the set functions and operators work properly.

MINIA UNIVERSITY

FACULTY OF COMPUTER & INFORMATION SCIENCES

Object-Oriented Programming

Assignment #4

**Part A:**

Define a base class***Person***, and two derived classes:***Professor*** and***Student***.

 The base class**Person** has two data members:**name** (a string) and**id** (an int), a

***constructor***, and a member function***displayPerson()*** to display the person name

and id.

 The derived class**Student** has a data member**nPapers** (an int represents the no.

of research papers published by the student and his/her supervisor), a

***constructor***, and a member function:***papers()*** that returns the no. of papers.

 The derived class**Professor** has two data members:**StudentsList** (an array of

**Student** objects that holds the students supervised by the professor) and

**nStudents** (an int represents the no. of those students), a***constructor***, and three

member functions:***registerStudent()*** that accepts a**Student** object and enters it in

the array**StudentsList**,***total()*** that calculates and returns the total no. of papers

published by the professor and his students, and***displayStudents()*** that displays

the data of the students supervised by the professor.

**Part B:**

Write a main program that reads a professor’s name and id, and creates a**Professor**

object for that professor. Then, the program reads a series of students’ names, ids,

and no. of papers, until the name XXX is entered. For each student, a**Student** object

is created and entered to the array**StudentsList** of the**Professor** object by

registering that student with the professor. After all students have been entered, the

data of the students supervised by the professor are displayed, then the total no. of

papers published by the professor and his students is calculated and displayed.

MINIA UNIVERSITY

FACULTY OF COMPUTER & INFORMATION SCIENCES

Object-Oriented Programming

Assignment #5

**Part A:**

 Define a class***Node*** that represents a node of a linked list. It has two data

members:***data*** (of type int) and***next*** (a pointer to next node), a constructor, and 5

member functions: SetData(), GetData(), NextNode(), InsertAfter(), and

DeleteAfter().

 Define a base class***LnkLst*** that represents a linked list. It has two data members:

***head*** (a pointer to the list head) and***size***, a constructor, and 10 member functions:

InsertRear(), InsertFront(), DeleteFront(), DeleteRear(), PrintLst(), FindItem(),

ClearList(), Empty(), ListSize(), and DeleteItem().

 Define a child class cStack of class LnkLst that has a constructor, and 2 member

functions: Push(), and Pop().

 Define a child class cQueue of class LnkLst that has a constructor, and 2 member

functions: Enqueue(), and Dequeue().

 Define a child class cDeque of both classes: cStack and cQueue. It has a

constructor, and 4 member functions: Insert\_Right(), Insert\_Left(),

Remove\_Left(), and Remove\_Right().

**Part B:**

Write a main program that tests all the operations of the 4 classes: LnkLst, cStack,

cQueue, and cDeque. The program prompts the user to enter the type of the object (L,

S, Q, or D) to be created, and creates the required object, then tests all the operations

that can be performed on it.

MINIA UNIVERSITY

*x*1 *x* 2     

*y*1*y*2

FACULTY OF COMPUTER & INFORMATION SCIENCES

Object-Oriented Programming

Assignment #6

(a) Declare and implement a class***Point***, that has two data members:***x*** and***y***,

representing the x- and y-coordinates of a point, and has 3 member functions:

***getPoint()*** that reads the values of x and y of a point object;***showPoint()*** the

displays the values of x and y of a point object; and***distance()*** that calculates

the distance between two points (x1,y1) and (x2,y2) using the formula:

2 2

Distance =

(b) Declare and implement the following class hierarchy:

(i) An abstract base class***Shape*** that has 3 pure virtual member functions:

***getdata()***,***showdata()***, and***getArea()***.

(ii) A derived class***Circle*** (from class***Shape***) that represents a circle. It has 2

data members: one of type**Point** representing the center of the circle, and

one of type**double** representing the radius of the circle, and implements the

3 member functions:***getdata()***, to read the center of the circle;***showdata()***, to

displays the circle data; and***getArea()***, to calculate the area of the circle.

(iii) A derived class***Quadr*** (from class***Shape***) that represents a quadrilateral. It

has 5 data members: 4 of type**Point** representing the 4 endpoints of the

quadrilateral, and one of type**double** representing the area of the

quadrilateral, and implements the 3 member function:***getdata()***, to read the 4

endpoints of the quadrilateral;***showdata()***, to display the quadrilateral data;

and***getArea()***, to calculate the area of the quadrilateral as the sum of the

areas of the two triangles forming it. The area of a triangle of sides a, b, and c

is obtained by the formula: area = *s*(*s* *a*)(*s* *b*)(*s* *c*) , where

*s* 

*a* *b* *c*

2   .

(iv) A derived class***Rectangle*** (from class***Quadr***) that has two additional data

members:***length*** and***width*** of type**double**. It redefines the function

***getArea()*** to calculate the area of the rectangle as follows: area = length 

width, and the function***showdata()*** to display all the data of the rectangle. It

has two additional member functions:***getlength()*** and***getwidth()*** that

calculates the length and width of the rectangle.

(c) Write a main program that creates an array of 10 pointers to***Shape***. In a

loop, ask the user for data about a shape and its type (circle, quadrilateral, or

rectangle), and use**new** to create a suitable object (***Circle***,***Quadr***, or

***Rectangle***) to hold the data, and then put the pointer to the created object in

the array. When the user finishes entering the data for all shapes, display the

data of all the shapes entered.

MINIA UNIVERSITY

FACULTY OF COMPUTER & INFORMATION SCIENCES

Object-Oriented Programming

Assignment #7

Using Template Class Stack

**Part A:**

Using the Template Class Stack, write a function that implements the following

algorithm, which accepts an infix expression, and converts it to postfix (Reverse

Polish Notation (RPN)) form, then returns it.

For example, if the function accepts the infix expression

(6-(2+3))\*(3+8/2)+2

then the function should return its postfix form: 623+-382/+\*2+

Note that this algorithm uses a character stack.

Algorithm Infix\_To\_Posfix

Begin

Create an object S of char stack;

While items remain in infix expression Do

Begin

Get next item;

If item is an operand then

Add operand to postfix expression;

Else if item is a left parenthesis then

Push left parenthesis onto S;

Else if item is a right parenthesis then

Repeat

Pop operator from S;

If operator is not a left parenthesis then

Add operator to postfix expression;

Until operator is a left parenthesis;

Else *// item is an operator*

While not Empty (S) and top of S has priority  operator Do

Begin

Pop operator from S;

Add popped operator to postfix expression;

End While;

Push operator;

End If

End While

While not Empty (S) Do

Begin

Pop operator from S;

Add popped operator to postfix expression;

End While

Destroy S;

Return postfix expression

End.

**Part B:**

**P.T.O.**

Using the Template Class Stack, write a function that implements the following

algorithm, which accepts an RPN (postfix) expression, and evaluates it, then returns

the result.

For example, if the function accepts the postfix expression

623+-382/+\*2+

then the function should return its value: 9

Note that this algorithm uses an integer stack.

Algorithm Evaluate\_Postfix

Begin

Create an object S of int stack;

While items remain in postfix expression Do

Begin

Get next item;

If item is an operand then

Push item onto S;

Else *// item is an operator*

Pop operand into Op2;

Pop operand into Op1;

Case operator type Of

‘+’ : Result = Op1 + Op2;

‘-’ : Result = Op1 – Op2;

‘\*’ : Result = Op1 \* Op2;

‘/’ : Result = Op1 / Op2;

‘%’ : Result = Op1 % Op2;

End Case;

Push Result onto S;

End If

End While

Pop Result from S;

Return Result;

Destroy S;

End.

**Part C:**

Write a program that:

 Reads an arithmetic expression in infix form.

 Calls Function Infix\_To\_Posfix() to convert the input infix expression to postfix

form.

 Calls Function Evaluate\_Posfix() to evaluate the postfix expression returned from

Function Infix\_To\_Posfix().

 Displays the input infix expression, the postfix expression returned from Function

Infix\_To\_Posfix(), and the value returned from Function Evaluate\_Posfix().

MINIA UNIVERSITY

FACULTY OF COMPUTER & INFORMATION SCIENCES

Object-Oriented Programming

Assignment #8

Consider the following declaration of class String:

const int SIZE=50;

class String {

private:

char str[SIZE];

public:

String(); // no arguments constructor

String(char s[]); // one-argument constructor

void display(); // display the string

String operator + (String s); // add two strings

};

**Part A:**

 Write the implementation of the constructors and the member functions of class

String.

 Add an exception class, and throw an exception in the one-argument constructor if

the initialization string is too long. Throw another exception in the overloaded**+**

operator if the result will be too long when the two strings are concatenated. Use

an argument in the exception constructor to report which of these errors has

occurred.

**Part B:**

Write a main program that uses class String to perform the following tasks:

1. Read two strings from the keyboard, and display them.

2. Initialize two objects of class String with the input strings.

3. Create another object of class String, then add the two strings and assign the

result to this object.

4. Display the new string.

5. Ask the user whether he/she wants to continue, and if so go to task 1, otherwise

exit the loop.

Insert both the try block and the catch block in the appropriate places inside the loop

so that after an exception you can go back to the top of the loop, ready to ask the user

for more input.

Define a class**Array**, whose objects are arrays of integers. Class**Array** has two data

members:***elem*** (array of integers) that holds the elements of the array, and***size*** (of

type int) that holds the number of elements in the array. It uses a static member,

***arrayCount***, to keep track of the number of array objects that have been created. Each

time an object is created,***arrayCount*** is incremented by 1. Class**Array** has:

1. a**default constructor,** which receives a single int argument that represents the

size of the array object and has a default value of 10. It also initializes the

array elements to 0.

2. a**copy constructor** that initializes an array object by making a copy of an

existing array object.

3. a**destructor** that sets the size of an array object to 0, and decrements the

number of objects by 1.

4. a function**getSize()** that returns the size of an array object.

5. a static function**getArrayCount()** that returns the number of array objects.

6. a set of operators: an**assignment operator =** that copies the elements of the

right hand side array object into the left hand side one and returns a reference

to the left hand side object; an**equality operator ==** that compares two array

objects and returns true if they have the same elements, an**inequality**

**operator !=**; a**subscript operator []** that returns a reference to a specified

element of an array object if the index value is in the range, otherwise it

displays an error message and exits; an**insertion operator <<** that prints the

elements of an array object on the specified output stream; and an**extraction**

**operator >>** that inputs the elements of an array object from the specified

input stream.

**Question 2:**

Define a class**account** that represents a bank account and has three data members:

account number**accNo** (of type int), current balance**balance** (of type double), and

periodic interest rate**rate** (of type double). It also has:

(a) a constructor that creates an account object and sets its data members to given

values.

(b) a member function**show\_account()** that displays the information of an

account object.

(c) a member function**deposit()** that accepts an amount of money, then increases

the balance of an account object by this amount.

(d) a member function**withdraw()** that accepts an amount of money, then

decreases the balance of an account object by this amount if it does not exceed

the balance and returns the amount withdrawn; otherwise, returns 0.

(e) a member function**compound()** that computes the interest of an account

object and adds it to its balance, where interest = balance x rate.

Then, write a program that performs the following tasks:

(i) Reads the information of 10 accounts from the keyboard, and create 10

**account** objects for these accounts, then

(ii) Asks the user to enter one of the following letters and performs the

corresponding operation:

D: Asks the user to enter an account number and an amount of money, and

deposits this amount in the specified account, then displays the account.

W: Asks the user to enter an account number and an amount of money, and

attempts to withdraw this amount from the specified account, then

displays a message indicating whether the withdrawal is done or not.

C: Computes the interest for the 10 accounts, then displays the accounts.

(iii) Repeats task (ii) until the user enters the letter X.