**C++ Workshop – 150018**

**Homework Assignment #2**

**Classes with Dynamic Memory**

**general:**

* Pay attention to readability and indentation, of program.
* Do exactly what you are asked for in each question.
* Define function according to need.
* Don't forget meaningful names for variables, document code and functions, and add running examples for your program at end.
* Homework is given in individually.
* You have one week to give in your homework.

**Question 1**

* 1. Write a class to represent points on a plane. You should include a method that computes the distance between two points.(you may use the same class you used in homework 1 as a base)
  2. Write a class that represents a planar polygon. The class should contain the following fields:
     + an array of vertices (points as defined in a.)
     + number of vertices in the polygon

Add the following methods to the class:

* + - constructor
    - copy constructor
    - destructor
    - method addPoint that receives an index and a vertex and adds the vertex to the array of vertices.
    - method to compute the perimeter of the polygon (sum of the distances between adjacent vertices).You can assume the vertices are ordered according to how they are on the polygon.
    - A Boolean method receiving a polygon and checks whether the current polygon is equal to the polygon received. Polygon's are defined equal when they have the same number of vertices and same values for the vertices (not necessarily in the same order in the array).  
      For example the polygon (0,0) (1,1) (2,0) is equal to (1,1) (2,0) (0,0).
  1. To help with understanding the constructor you must add printing in the constructor methods as following:  
     in the constructor: in constructor  
     copy constructor: in copy-constructor  
     destructor: in destructor
  2. Write a main program that reads in the data of two polygons and prints their perimeter rounded to the nearest integer value, In the following manner:  
      if they are equal print equal and their perimeter, otherwise print the perimeter for each one.
  3. For each polygon the program will print:  
     enter number of sides, and then enter the number,
  4. Then print enter the point values, and enter the point values.  
     The format of the input is  
     N x1 y1 x2 y2 … xN yN where N is the number of vertices and xi and yi are the coordinates of a vertex of the polygon. If there is an error in the input, print ERROR

Example (square of length 2): Example (3-4-5 triangle):

enter number of sides:  
3  
in constructor

enter the point values:

(10,10) (10,14) (13,10)  
enter number of sides:  
3  
in constructor

enter the point values:

(13,10) (10,10) (10,14)

in copy-constructor

in destructor

equal. perimeter: 12  
in destructor

in destructor

enter number of sides:  
4  
in constructor

enter the point values:

(0,0) (0,2) (2,2) (2,0)

enter number of sides:  
3  
in constructor

enter the point values:

(1,1) (2,0) (3,1)

in copy-constructor

in destructor

perimeter: 8

perimeter: 5  
in destructor

in destructor

Comment: In order to use math functions, include the library cmath

**Question 2**

Write a class **Vector** that implements a list of integers of any given size. The class consists of the following fields:

* + data – pointer to an array of integers
  + capacity – size of vector (maximum number of elements in list)
  + size – actual number of elements found in vector

In addition, the class should have the following methods:

* default constructor that receives an integer which is the capacity of the list, and creates the vector of that size. if no parameter received, then initializes the maximum number of elements of the list to 2.
* copy constructor
* destructor
* method **getCapacity()** which returns the maximum number of elements that the list can contain
* method **getSize()** which returns the actual number of elements in the vector
* method **print()** which prints the following: maximum size, actual size, and elements of the vector  
  For example, for a vector of size 3 containing the elements 5 and 19:

capacity: 3 size: 2 val: 5 19

* method named assign (acts like operator =) to assign one vector to another, gets another Vector as parameter.
* method names **isEqual()** (acts like operator == )that return true whenever two vectors are identical (in size and content) and otherwise false
* method named **at**(), that receives an index (acts like operator [] ) for assignment and retrieving a value in the vector. If the requested index is not in the range, then print ERROR and return first element. Prototype of method:

Int& at(int index)

* method **strcatcat**() which computes the scalar (dot) product of two vector operands (i.e., the sum of the product of corresponding elements in the vectors). If the vectors do not have the same number of elements it should print ERROR and return -1.
* method **strnewcat** . The method creates and returns a new vector whose contents is the concatenation of two vector operands
* method **clear()** which clears the **contents** of the vector
* method **delLast()** which deletes the last element of the vector. If the vector was empty, it prints **ERROR**
* method **insert(int val)** which appends the given val to the next open place in the vector. If the vector was full it expands the vector dynamically by making it twive the size, and copy the values etc, release the memory.

Use the following main program to check the methods that you wrote.

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

using namespace std;

enum options

{

stop, assignment, isEqual, mult, add, clear, delLast, at, insert

};

int main()

{

Vector v1(10), v2(10), v3;

for (int i = 1; i <= 4; i++)

{

v1.insert(i);

v2.insert(i + 4);

}

int choice, val, index;

cout << "enter your choice 0-8:\n";

cin >> choice;

while (choice)

{

switch (choice)

{

case assignment: v3.assign(v1);

break;

case isEqual: if (v1.isEqual(v2)) cout << "v1==v2\n"; else cout << "v1!=v2\n";

break;

case mult: cout << "v1\*v2=" << v1.strcatcat(v2) << endl;

break;

case add: v3.assign(v1.strnewcat(v2));

break;

case clear: v1.clear();

break;

case delLast: v2.delLast();

break;

case at: cout << "enter index:" << endl;cin >> index; cout << "enter value:" << endl; cin >> val;v3.at(index) = val;

break;

case insert: cout << "enter value:" << endl;cin >> val;v3.insert(val);

break;

default: cout << "ERROR";

}

v1.print(); v2.print(); v3.print();

cout << "enter your choice 0-8:\n";

cin >> choice;

}

return 0;

}

Examples

enter your choice 0-8:

2

v1\*v2=70

capacity: 10 size: 4 values: 1 2 3 4

capacity: 10 size: 4 values: 5 6 7 8

capacity: 2 size: 0 values:

enter your choice 0-8:

0

enter your choice 0-8:

4

capacity: 10 size: 4 values: 1 2 3 4

capacity: 10 size: 4 values: 5 6 7 8

capacity: 20 size: 8 values: 1 2 3 4 5 6 7 8

enter your choice 0-8:

0