Exercise 5:

Submit your solutions (source code) to the questions below by email to your instructor and TA(s) by Monday, November 5th (16:30).

Question 1: Overloading (10 points).

Create overloaded versions of the function dist() that computes the Euclidean distance bewteen two vectors (see Exercise 4, Question 2). Specifically, create the following overloaded versions:

- double dist(int*, int*, int);
- double dist(float*, float*, int);
- double dist(double*, double*, int);

To test your functions, create a file "test_utils.cpp" and type the following code in it:

```
// test_utils.cpp

#include <cassert>
#include <cmath>
// Include other needed headers here
```

```
int main(void) {
  const double epsilon = 1e-5;

  int intArray1[] = {0, 0, 0, 0, 0};
  int intArray2[] = {1, 1, 1, 1, 1};
  double d = dist(intArray1, intArray2, 5);
  assert(fabs(d-sqrt(5.0))<=epsilon);

  float fltArray1[] = {0.0f, 0.0f, 0.0f};
  float fltArray2[] = {1.0f, 1.0f, 1.0f};
  d = dist(fltArray1, fltArray2, 3);
  assert(fabs(d-sqrt(3.0))<=epsilon);

  double dblArray1[] = {1.0, 2.0};
  double dblArray2[] = {2.0, 1.0};
  d = dist(dblArray1, dblArray2, 2);
  assert(fabs(d-sqrt(2.0))<=epsilon);

  return 0;
}
```

Question 2: Constructor and destructor (40 points).

In the following 3 questions, you will exercise your understanding of how constructor, copy constructor, assignment operator and destructor work. Create a file named "ArrayStack.h" that will contain the definition of the class ArrayStack. ArrayStack is an implementation for a

LIFO (Last-in-first-out) stack using a resizing array to hold elements of type int. Type and complete the following code:

```
// ArrayStack.h

#include <string>
#include <iostream>

#ifndef ARRAY_STACK_H
#define ARRAY_STACK_H

class ArrayStack {
 private:
  int N; // number of items in the stack
  int* items; // stack items
  int allocSize; // size of memory allocated

  void resize(int max) {
   // Move stack to a new array of size max
   allocSize = max;
   int* temp = new int[max];
   // Copy
   for (int i = 0; i < N; ++i) {
     temp[i] = items[i];
   }
   delete[] items;
   items = temp;
  }

 public:
```

```
// Constructors:
ArrayStack()
    /* COMPLETE ... init N to 0,
     * allocate memory for an array of size one and make
     * items point to it */

explicit ArrayStack(int allocSz)
    /* COMPLETE ... init N to 0,
     * pre-allocate memory for an array of size allocSz
     * and make items point to it */

// Destructor::
~ArrayStack() {
  // COMPLETE
}

void push(int item) {
  if (N == allocSize) resize(2*allocSize);
  items[N++] = item;
}

int pop() {
  int ret = items[--N];
  if (N > 0 && N == allocSize/4) resize(allocSize/2);
  return ret;
}

bool isEmpty() const { return N == 0; }

int size() const { return N; }
```

```
};

#endif // ARRAY_STACK_H
```

In order to test this class, type the following code in a file named "test_ArrayStack.cpp":

```
// test_ArrayStack.cpp
#include "ArrayStack.h"

int main(void) {
  // Create an instance of ArrayStack named stack1
  //   using the default constructor
  // Push 1,2,3,4,5 in this stack

  // Create another instance of ArrayStack named stack2
  //   using the other constructor, specify an original size of 5
  // Push 1,2,...,10 in this stack

  return 0;
}
```

Question 3: Copy constructor (25 points).

Please complete the definition of the class ArrayStack by adding a copy constructor. The copy constructor should carefully manage the owned data (i.e. the pointers). In the copy constructor, please also add a line (of information) that prints: "Copy Constructor". You can

add the copy constructor directly in the file "ArrayStack.h". In order to test your class, edit the file "test_ArrayStack.cpp" by adding the following code in the main function:

```
int main(void) {
// ...
// here is the code from question 2
// ...

 ArrayStack stack3(stack1);
 ArrayStack stack4 = stack2;

 return 0;
}
```

Question 4: Assignment operator (25 points).

To complete the class ArrayStack, add an assignment operator. Please edit the file "ArrayStack.h" and add the code for the assignment operator. Please be careful to handle self-assignment correctly. The assignment operator should carefully manage the owned data (i.e. the pointers). In the assignment operator, please also add a line (of information) that prints: "Assignment Operator". In order to test your class, edit the file "test_ArrayStack.cpp" from the previous question by adding the following code in the main function:

```
int main(void)
{
// ...
```

```
// here is the code from the previous questions
// ...

 ArrayStack stack5;
 stack5 = stack1;
 ArrayStack stack6(10);
 stack6 = stack6;

 return 0;
}
```