# Exercise 3:

Submit your solutions (source code) to the questions below by email to your instructor and TA(s) by Monday, October 22nd (16:30).

## Question 1: Pointers (40 points).

Create a file: "test_pointers.cpp" then complete the following code according to the explanations in comments:

```
// test_pointers.cpp

#include <iostream>

int main(void)
{
  //
  // 1. declare a variable dblPtr as a pointer to a double
  // 2. allocate memory on the heap corresponding to one double and make dblPtr points to this location
  // 3. store the value 3.14 at the memory location pointed to by dblPtr
  // 4. print out the content of the memory location pointed to by dblPtr
  // 5. delete the memory pointed to by dblPtr

  //
  int a[5] = {1, 2, 3, 4, 5};
  // 6. declare a variable intPtr as a pointer to int
  // 7. make intPtr points to the beginning of the array a
  // 8. print out what intPtr points to and the content of a[0]
  // 9. increase intPtr by 1 and check that what it points to corresponds to a[1]
  // 10. make intPtr points to the fourth element of the array by increasing it by 2
  //     and check that what it points to corresponds to a[3]

  //
  int n = 10;
  // 11. declare a variable "fltArray" as a pointer to float and
  // make it point to an array of "n" element of type "float" created on the heap
  // 12. store in each fltArray[i] (for i=0 to n-1) the value float(i) / 2.0f;
  // 13. print out each element of fltArray
  // 14. delete the previously allocated memory

  //
  int m = 5;
  double** dblArray;
  // 14. allocate memory for a 2d array of size m * n on the heap (i.e. m arrays of size n).
  // Make dblArray points to this 2d array.
  for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
      dblArray[i][j] = (3.14 * i) / (j+1.0);

  // 15. print out each element of dblArray
  // 16. delete the memory allocated for the 2d array

  return 0;
}
```

## Question 2: Pass by reference and by pointer: Swap (30 points).

Step 1: Create the files: "swap.h", "swap.cpp" and "test_swap.cpp". In "swap.h" write the prototype (i.e. the function declaration) for the function "swap()" that takes two arguments of type 'int' and swap their contents. In "swap.cpp" write the implementation of the function "swap()". In "test_swap.cpp" type and complete the following code:

```
// test_swap.cpp

// <- Include all the necessary headers here

int main(void) {
  int i = 1;
  int j = 2;
  swap(i, j);
  std::cout << "i=" << i << " and j=" << j << std::endl;
  return 0;
}
```

Step 2: Create the files: "swap_ptr.h", "swap_ptr.cpp" and "test_swap_ptr.cpp". In "swap_ptr.h" write the prototype for a function "swap_ptr()" that takes two pointers to 'int' and swap the content of the memory pointed by these pointers. In "swap_ptr.cpp" write the implementation of the function "swap_ptr()" declared in the header "swap_ptr.h". In "test_swap_ptr.cpp" type and complete the following code:

```
// test_swap_ptr.cpp"

// <- Include all the necessary headers here

int main(void) {
  int i = 1;
  int j = 2;
  /* call swap_ptr() here in order to swap the contents of i and j */
  std::cout << "i=" << i << " and j=" << j << std::endl;
  return 0;
}
```

## Question 3: Pass by reference and by pointer: Replace (30 points).

Consider the following function replace_all:

```
void replace_all(string& in, char search, char replace_with) {
  for (unsigned int i = 0; i < in.length(); ++i) {
    if (in.at(i)==search) {
      in.at(i) = replace_with;
    }
  }
}
```

In the code above, the method at(i) allows to access the char at position i in the string. This function can be tested with the following code (you can save the code in a file named test_replace_all.cpp, do not forget to add the necessary includes):

```
int main(void) {
  string str1 = "uaizu";
  replace_all(str1, 'u', 'e');
  assert(str1=="eaize");

  string str2 = "programming";
  replace_all(str2, 'm', 'r');
  assert(str2=="prograrring");

  return 0;
}
```

Rewrite the function replace_all using pass by pointer instead of pass by reference for the first argument (the input string). Modify the main function to test your modified "replace_all" function accordingly.