Jeongbin Son
CS403
Assignment 4

Question 1

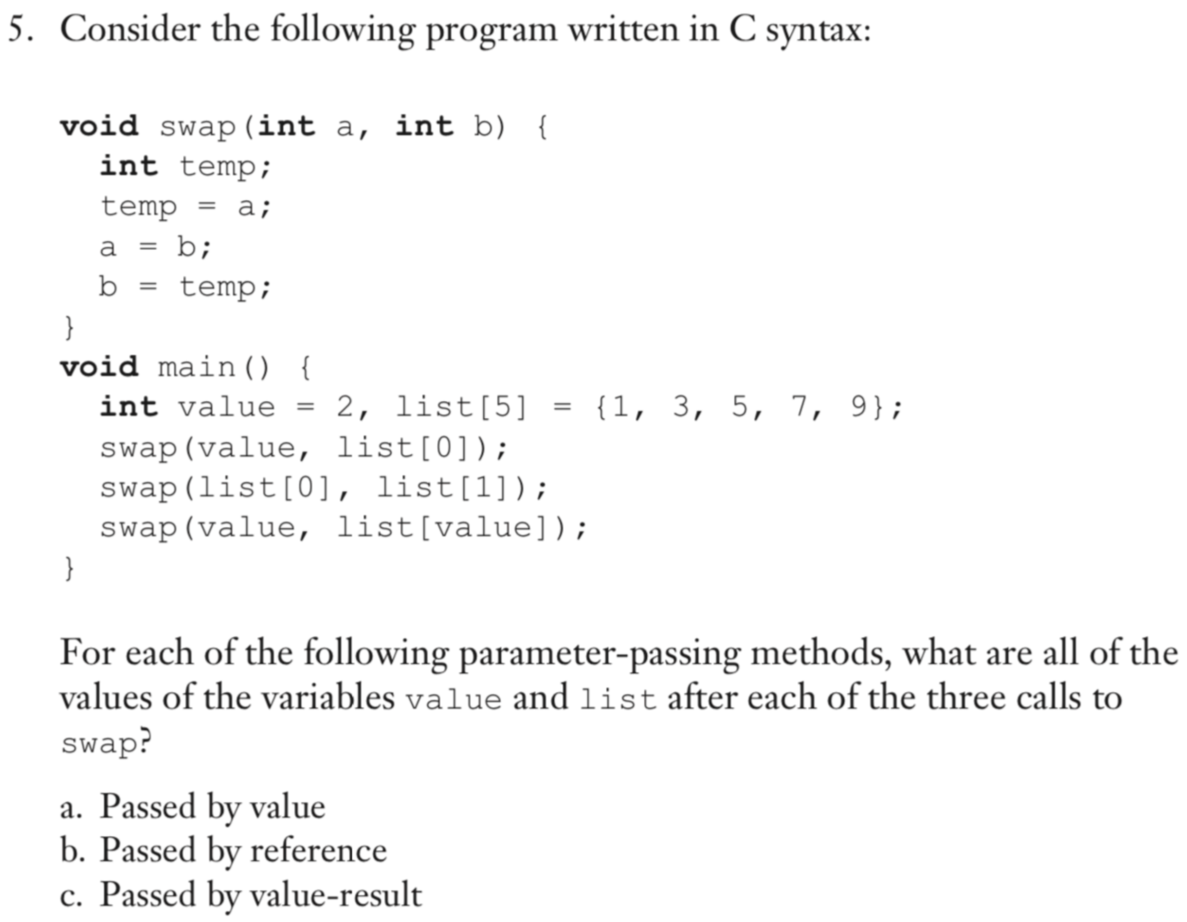
1. Write a program in C that uses the `qsort` function to sort an array of real numbers (`double`). Paste below only the comparison function passed to `qsort` as a parameter.

Answer:

```
int compare_doubles(const void* a, const void* b) {
   const double* da = (const double*) a;
   const double* db = (const double*) b;

   if (*da > *db) return 1;
   else if (*da < *db) return -1;
   else return 0;
}
```

```
int compare(const void *a, const void *b) {
        double x = *(const double *)a;
        double y = *(const double *)b;

        if (x < y) return -1;
        if (x > y) return 1;
        return 0;

}
```

Question 2



5. Consider the following program written in C syntax:

```
void swap(int a, int b) {
  int temp;
  temp = a;
  a = b;
  b = temp;
}
void main() {
  int value = 2, list[5] = {1, 3, 5, 7, 9};
  swap(value, list[0]);
  swap(list[0], list[1]);
  swap(value, list[value]);
}
```

For each of the following parameter-passing methods, what are all of the values of the variables `value` and `list` after each of the three calls to `swap`?

a. Passed by value
b. Passed by reference
c. Passed by value-result

A. Passed by value:

swap(value, list[0]) -> no change, {1, 3, 5, 7, 9}
swap(list[0], list[1]) -> no change {1, 3, 5, 7, 9}
swap(value, list[value]) -> no change {1, 3, 5, 7, 9}
value is still 2, list is still {1, 3, 5, 7, 9}

B. Passed by reference:

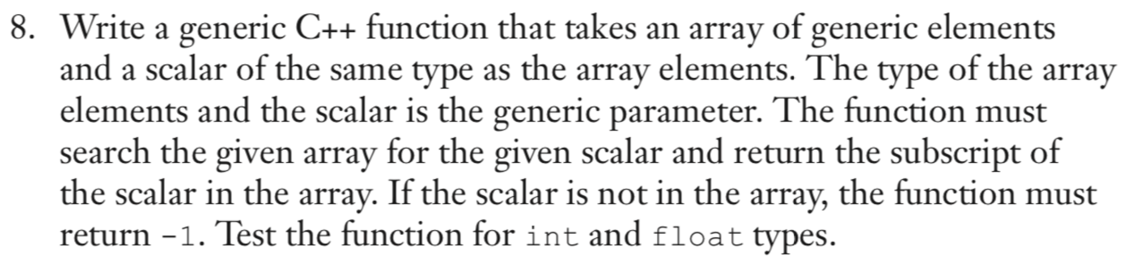
swap(value, list[0]) -> value = 1, list = {2, 3, 5, 7, 9}
swap(list[0], list[1]) -> value = 1, list = {3, 2, 5, 7, 9}
swap(value, list[value]) -> value = 2, list = {3, 1, 5, 7, 9}

C. Passed by value-result:

swap(value, list[0]) -> value = 1, list = {2, 3, 5, 7, 9}

```
swap(list[0], list[1]) -> value = 1, list = {3, 2, 5, 7, 9}
swap(value, list[value]) -> value = 2, list = {3, 1, 5, 7, 9}
```

Question 3

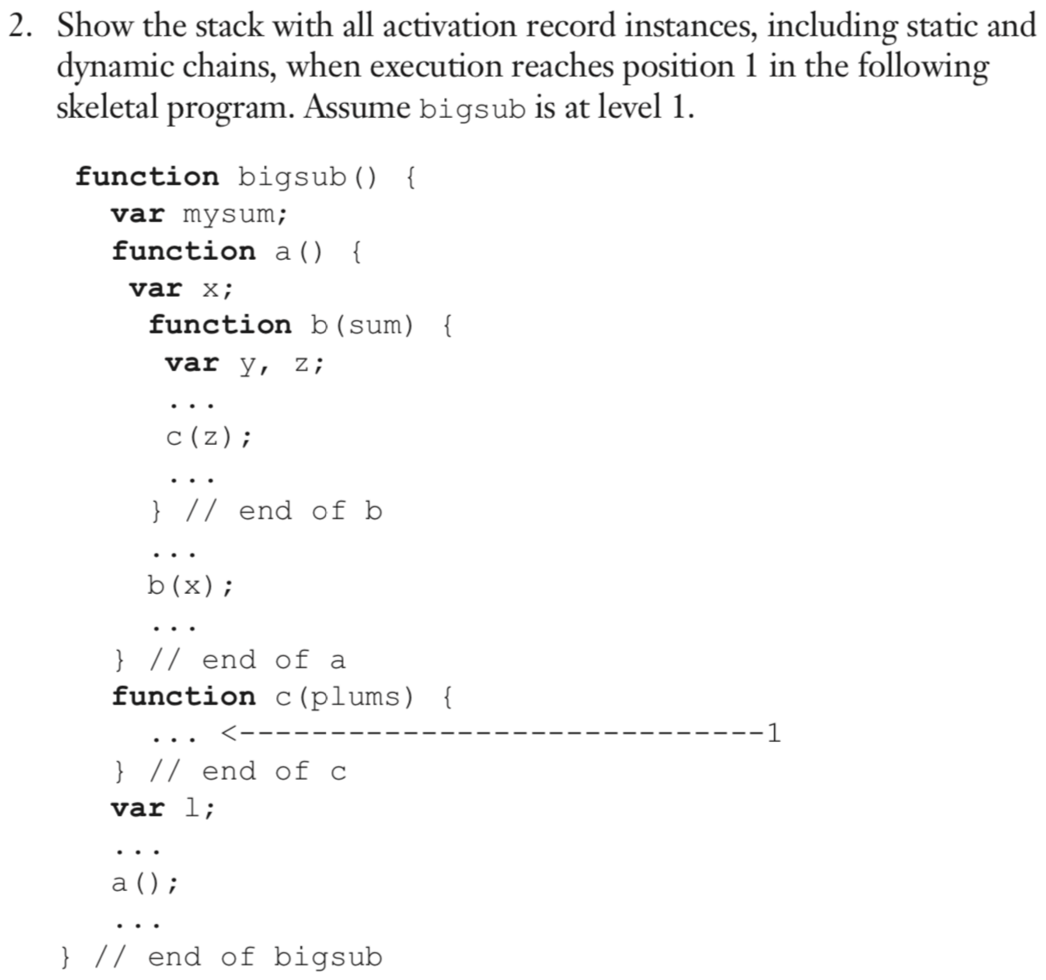


8. Write a generic C++ function that takes an array of generic elements and a scalar of the same type as the array elements. The type of the array elements and the scalar is the generic parameter. The function must search the given array for the given scalar and return the subscript of the scalar in the array. If the scalar is not in the array, the function must return `-1`. Test the function for `int` and `float` types.

```
#include <iostream>

template <typename T>
int findIndex(const T* array, size_t size, T scalar) {
   for (size_t i = 0; i < size; ++i) {
      if (array[i] == scalar) {
         return i;
      }
   }
   return -1; // If not found
}

int main() {
   int intArray[] = {1, 2, 3, 4, 5};
   float floatArray[] = {1.1f, 2.2f, 3.3f, 4.4f, 5.5f};

   // Test with int
   int intIndex = findIndex(intArray, 5, 3);
   std::cout << "Index in intArray: " << intIndex << std::endl;

   // Test with float
   int floatIndex = findIndex(floatArray, 5, 3.3f);
   std::cout << "Index in floatArray: " << floatIndex << std::endl;
```

```
    return 0;
}
```

Question 4



2. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. Assume `bigsub` is at level 1.

```
 function bigsub() {
   var mysum;
   function a() {
    var x;
     function b(sum) {
      var y, z;
      ...
      c(z);
      ...
     } // end of b
     ...
     b(x);
     ...
   } // end of a
   function c(plums) {
     ... <----------------------------1
   } // end of c
   var l;
   ...
   a();
   ...
} // end of bigsub
```

Stack:

```
|--------------------------------------------------|
| Activation Record for c                          |
| Dynamic Link: points to Activation Record for b  |
| Static Link: points to Activation Record for bigsub (level 1) |
| Local Variables: plums (parameters, etc.)        |
|-------------------------------------------------|
| Activation Record for b                          |
| Dynamic Link: points to Activation Record for a  |
| Static Link: points to Activation Record for bigsub (level 1) |
| Local Variables: y, z                           |
|-------------------------------------------------|
| Activation Record for a                          |
| Dynamic Link: points to Activation Record for bigsub |
| Static Link: points to Activation Record for bigsub (level 1) |
| Local Variables: x                              |
|-------------------------------------------------|
| Activation Record for bigsub                     |
| Dynamic Link: points to the base of the stack or null if it's the first call |
| Static Link: null (as it is at level 1, the topmost level) |
| Local Variables: mysum, i                        |
|-------------------------------------------------|
```