מת"מ ת"ב 3 – חלק יבש

:2.1

```
/**
 * Get the length of the biggest collection(starting from the beginning) that
 * passes the "check_collection" test
 *
 * @param1 begin the iterator pointing to the start of the collection
 * @param2 begin the iterator pointing to the end of the collection
 * @param3 check_collection a function object containing a function that
checks
 * if a collection passes a test
 * @return
 * the length of the biggest collection(starting from "begin") that passes
the
 * test(if no collection passes the test then return 0)
 */
template<typename Iterator, class CheckCollection>
int filterLength(Iterator begin, Iterator end,
                        CheckCollection& check_collection) {
      while(begin != end) {
            if(check_collection(begin, end)) {
                  return (end - begin + 1); //return length of collection
            }
            --end;
      }
      //begin = end
      if(check_collection(begin, end)) {
            return (end - begin + 1); //return length of collection
      }
      else { //return the length of an empty collection
            return 0;
      }
}
```

ניתן להשתמש באלגוריתם שכתבנו כדי למצוא במערך של מספרים שלמים את האיבר הראשון שגדול יותר מהאיבר הבא אחריו. אם אין כזה, יש להדפיס הודעה מתאימה

```
/**
 * A function object class that checks if a collection is in ascending order
 */
class IsAscending {
public:
	/**
	 * Check if a collection(from "begin" to "end") is in ascending order
	 *
	 * @param1 begin the iterator pointing to the start of the collection
	 * @param2 begin the iterator pointing to the end of the collection
	 * @return
	 * true if the collection is in ascending order, false otherwise
	 */
	template<class Iterator>
	bool operator()(Iterator begin, Iterator end) {
		if (begin == end) { //if the collection is of size 1
			return true;
		}
		Iterator left = begin;
		Iterator right = ++begin;
		while(left != end) { //loop on all close pairs in the collection
			if (*left > *right) {
				return false;
			}
			++left;
			++right;
		}
		return true;
	}
};


//an example of using filterLength
int main() {
	const int n = 11;
	int arr[n] = {1, 2, 4, 7, 10, 14, 3, 6, 12, 24, 48};
	IsAscending is_ascending;
	int length = filterLength(arr, arr + (n-1), is_ascending);
	if(length == n) { //if the entire array is ascending
		cout << "No element in array that is bigger then next element" <<
endl;
	}
	else {
		cout << arr[length - 1] << endl;
	}
	return 0;
}
```

:2.2

```
class A {
public:
	virtual void printError() {
		cout << "The copied class will print this" << endl;
	}
	void printError() const {
		cout << "The const class will print this" << endl;
	}
};

class B : public A {
	void printError() override{
		cout << "The reference class will print this" << endl;
	}
};

//throws the object B
void f() {
	throw B();
}
```