Exercise: srcFacts Concerns Section: Email: Signed Name:

Allowed Concerns: srcFacts, srcML, XML, Parsing

Concern	srcfacts.cpp#L445-L485
Parser	445 `// parse start tag`
Parser	446 `assert(content.compare(0, "<"sv.size(), "<"sv) == 0);`
Parser	447 `content.remove_prefix("<"sv.size());`
Parser	449 `if (content[0] == ':') {`
Parser	449 `    std::cerr << "parser error : Invalid start tag name\n";`
Parser	450 `    return 1;`
Parser	451 `}`
Parser	452 `std::size_t nameEndPosition = content.find_first_of(NAMEEND);`
Parser	453 `if (nameEndPosition == content.size()) {`
Parser	454 `    std::cerr << "parser error : Unterminated start tag '" << content.`
Parser	455 `    return 1;`
Parser	456 `}`
Parser	457 `size_t colonPosition = 0;`
Parser	458 `if (content[nameEndPosition] == ':') {`
Parser	459 `    colonPosition = nameEndPosition;`
Parser	460 `    nameEndPosition = content.find_first_of(NAMEEND, nameEndPosition +`
Parser	461 `}`
XML	462 `const std::string_view qName(content.substr(0, nameEndPosition));`
Parser	463 `if (qName.empty()) {`
Parser	464 `    std::cerr << "parser error: StartTag: invalid element name\n";`
Parser	465 `    return 1;`
Parser	466 `}`
XML	467 `[[maybe_unused]] const std::string_view prefix(qName.substr(0, colonPo`
XML	468 `const std::string_view localName(qName.substr(colonPosition ? colonPos`
XML	469 `TRACE("START TAG", "qName", qName, "prefix", prefix, "localName", loca`
srcML	470 `bool inEscape = localName == "escape"sv;`
srcML	471 `if (localName == "expr"sv) {`
srcFacts	472 `    ++exprCount;`
srcML	473 `} else if (localName == "decl"sv) {`
srcFacts	474 `    ++declCount;`
srcML	475 `} else if (localName == "comment"sv) {`
srcFacts	476 `    ++commentCount;`
srcML	477 `} else if (localName == "function"sv) {`
srcFacts	478 `    ++functionCount;`
srcML	479 `} else if (localName == "unit"sv) {`
srcFacts	480 `    ++unitCount;`
srcML	481 `} else if (localName == "class"sv) {`
srcFacts	482 `    ++classCount;`
srcML	483 `}`
Parser	484 `content.remove_prefix(nameEndPosition);`
Parser	485 `content.remove_prefix(content.find_first_not_of(WHITESPACE));`

Allowed Concerns: srcFacts, srcML, XML, Parsing		
Concern	srcfacts.cpp#L295-L304	
Parser	295	`// refill content preserving unprocessed`
Parser	296	`int bytesRead = refillContent(content);`
Parser	297	`if (bytesRead < 0) {`
Parser	298	`    std::cerr << "parser error : File input error\n";`
Parser	299	`    return 1;`
Parser	300	`}`
Parser	301	`if (bytesRead == 0) {`
Parser	302	`    doneReading = true;`
Parser	303	`}`
Parser	304	`totalBytes += bytesRead`
Concern	srcfacts.cpp#L560-L568	
Parser	560	`std::size_t valueEndPosition = content.find(delimiter);`
Parser	561	`if (valueEndPosition == content.npos) {`
Parser	562	`    std::cerr << "parser error : attribute " << qName << " missing del`
Parser	563	`    return 1;`
Parser	564	`}`
XML	565	`const std::string_view value(content.substr(0, valueEndPosition));`
srcML	566	`if (localName == "url"sv)`
srcFacts	567	`    url = value;`
XML	568	`TRACE("ATTRIBUTE", "qname", qName, "prefix", prefix, "localName", loca`
Concern	srcfacts.cpp#L329-L336	
Parser	329	`// parse character non-entity references`
Parser	330	`assert(content[0] != '<' && content[0] != '&')`
Parser	331	`std::size_t characterEndPosition = content.find_first_of("<&")`
XML	332	`const std::string_view characters(content.substr(0, characterEndPositi`
XML	333	`TRACE("CHARACTERS", "characters", characters)`
srcFacts	334	`loc += static_cast<int>(std::count(characters.cbegin(), characters.cen`
srcFacts	335	`textSize += static_cast<int>(characters.size())`
Parsing	336	`content.remove_prefix(characters.size())`