

UNIVERSITY OF TORONTO
Faculty of Arts and Science
Midterm Exam – July 2007
CSC 309 H1 F
Instructor – Dr. Radu Negulescu
Duration – 1 hour
Examination Aids: One single-sided page containing notes

NAME _____

STUDENT NUMBER _____

PART I (Short Answers): _____ /20

PART II (Programming): _____ /40

Total: _____ /60

Part I of this examination is a series of short-answer questions.

Part II of this examination is a programming exercise.

Answer all questions in the spaces provided on this examination paper. There is no need to use more space than is provided. If you must, then use the back of the examination paper and so indicate in your answer.

General Advice:

- Skim through the entire exam before beginning your detailed work, to get a sense of where best to spend your time; if you get stuck on one question, go on to another and return to the difficult question later.
- For the programming exercise, if unsure of API details or language syntax details, try your best and include a comment indicating what you are attempting to achieve assuming a typical API.
- Partial credit will be granted in cases that demonstrate correct reasoning, even if an error leads to an incorrect final result.

Good luck!

Part I – Short Answer Questions

1.- [10 points] Describe 2 main advantages of JavaScript over Java Applets and 2 main advantages of Java Applets over JavaScript. (1 line each – maximum)

2.- [10 points] Draw a simple flowchart showing the main steps and the flow for an iterative process used in web development. Briefly describe a main reason why iterations are important.

Part II - Programming Question

3.- [40 points]

(a) [20 points] Use XHTML to write a web page with a form for ordering pizza. When loaded, the page should appear as shown below. To avoid clutter, use only 3 options for each select element (for example, use only months 07, 08, and 09 for the expiry date). Be sure to preserve alignment of items. The Submit button sends all the user inputs to URL <http://www.pizzaprontoofortoronto.ca/orders>. The Reset button resets the form. Write your solution in the space provided on Pages 3-4 starting at the DOCTYPE declaration below.

Pizza Pronto

Size

Topping

How many

Phone number

Address 1

Address 2

First name

Last name

Card #

Expiry

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

</html>

(b) [10 points] Use CSS to give a red color to the “Pizza Pronto” header and a green color to the backgrounds of the buttons, text inputs, and drop down lists, as shown below. The CSS code will be embedded in the web page of Part (a). Do NOT rewrite your solution to Part (a); write only the required CSS code in the space provided below and indicate the line number in your solution to Part (a) at which this CSS code will be inserted.

Pizza Pronto

Size

Topping

How many

Phone number

Address 1

Address 2

First name

Last name

Card #

Expiry

(c) [10 points] Modify your solution to Part (a) by adding a button labelled “More” to insert another pizza ordering section into the form. Pressing “More” repeatedly inserts multiple pizza ordering sections into the form. The ordering page area, located in between the title and card areas, may therefore comprise multiple ordering sections and a single “More” button as shown on Page 7. Use JavaScript to modify the DOM of the page so that the new inputs and selections will be included in the generated request.

Do NOT rewrite your solution to Part (a). Write in the space below only the new XHTML and JavaScript code required for Part (c), and indicate the line numbers of your solution to Part (a) where this new code will be inserted. Depending on the solution chosen, you may use one or more of the JavaScript methods and properties on Pages 8-9. You may also use other JavaScript methods and properties at your discretion. JavaScript also supports Java-like String operators such as +, +=, etc.

New aspect for the ordering page area before pressing "More":

Size

Topping

How many

New aspect for the ordering page area after pressing "More" twice:

Size

Topping

How many

Size

Topping

How many

Size

Topping

How many

Selected JavaScript reference

The `indexOf()` method returns the position of the first occurrence of a specified string value in a string, or -1 if no occurrences are found.

Syntax

```
stringObject.indexOf(searchvalue, fromindex)
```

Parameter	Description
searchvalue	Required. Specifies a string value to search for
fromindex	Optional. Specifies where to start the search

The `replace()` method is used to replace some characters with some other characters in a string.

Syntax

```
stringObject.replace(findstring, newstring)
```

Parameter	Description
findstring	Required. Specifies a string value to find. To perform a global search add a 'g' flag to this parameter and to perform a case-insensitive search add an 'i' flag
newstring	Required. Specifies the string to replace the found value from findstring

The `substring()` method extracts the characters in a string between two specified indices.

Syntax

```
stringObject.substring(start, stop)
```

Parameter	Description
Start	Required. Where to start the extraction. Must be a numeric value
Stop	Optional. Where to stop the extraction. Must be a numeric value

The `length` property returns the number of characters in a string.

Syntax

```
stringObject.length
```

The `innerHTML` property sets or returns the text of an element.

Syntax

```
elementObject.innerHTML=text
```

The cloneNode() method creates an exact copy of a specified node.

This method returns the cloned node.

Syntax

```
elementObject.cloneNode(include_all)
```

Parameter	Description
include_all	Required. If the Boolean parameter is set to true, the cloned node clones all the child nodes of the original node as well

The appendChild() method adds a node after the last child node of the specified element node.

This method returns the new child node.

Syntax

```
elementObject.appendChild(node)
```

Parameter	Description
Node	Required. The node to append

The getElementById() method returns a reference to the first object with the specified ID.

Syntax

```
document.getElementById(id)
```