

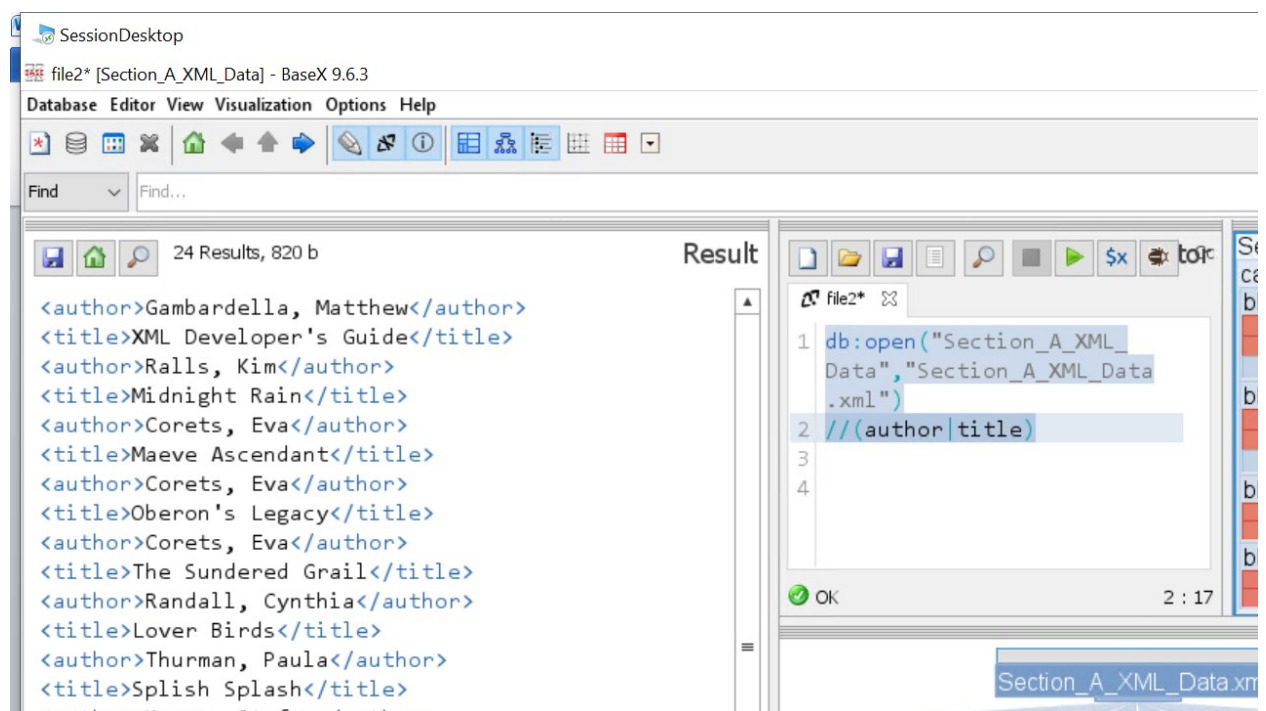
Assignment 3	Course : CIND110
Student : Samaraweera Wijesooriya	Date : August 8, 2022

Section A

1. [4 Pts.] Write an XPath expression to find all authors along with their corresponding books.

```
db:open("Section_A_XML_Data","Section_A_XML_Data.xml")
```

```
//*[author|title]
```



2. [4 Pts.] Write an XPath expression to find the prices of all books and their genre.

```
db:open("Section_A_XML_Data","Section_A_XML_Data.xml")
```

```
//*[price|genre]/text()
```

The screenshot shows the BaseX 9.6.3 interface. The main window is titled 'file2* [Section_A_XML_Data] - BaseX 9.6.3'. The menu bar includes Database, Editor, View, Visualization, Options, and Help. The toolbar contains various icons for file operations and editing. The 'Find' bar is visible. The 'Result' pane on the left shows 24 results, 193 b. The results are listed as follows:

Computer
44.95
Fantasy
5.95
Fantasy
5.95
Fantasy
5.95
Fantasy
5.95
Romance
4.95
Romance
4.95

The 'Editor' pane on the right shows the following XPath query:

```
1 db:open("Section_A_XML_Data", "
2 Section_A_XML_Data.xml")
3 //*(price|genre)/text()
4
```

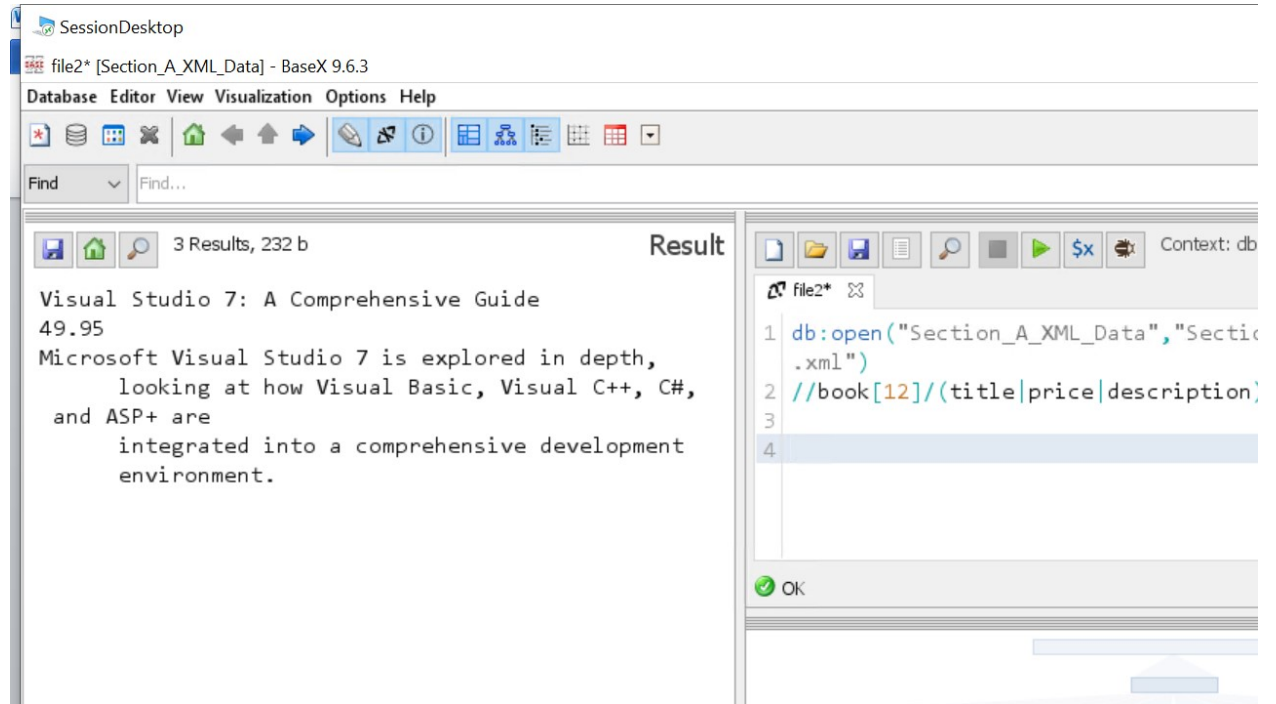
The 'Catalog' pane on the right shows a table with columns 'catalog', 'A_XML_Data', and 'bk101'. The table contains data for books bk101 through bk109, including their prices and genres.

catalog	A_XML_Data	bk101
bk101	bk102	bk103
bk102	bk104	bk105
bk103	bk106	bk107
bk104	bk108	bk109
bk105	bk110	bk111
bk106	bk112	bk113
bk107	bk114	bk115
bk108	bk116	bk117
bk109	bk118	bk119
bk110	bk120	bk121
bk111	bk122	bk123
bk112	bk124	bk125
bk113	bk126	bk127
bk114	bk128	bk129
bk115	bk130	bk131
bk116	bk132	bk133
bk117	bk134	bk135
bk118	bk136	bk137
bk119	bk138	bk139
bk120	bk140	bk141
bk121	bk142	bk143
bk122	bk144	bk145
bk123	bk146	bk147
bk124	bk148	bk149
bk125	bk150	bk151
bk126	bk152	bk153
bk127	bk154	bk155
bk128	bk156	bk157
bk129	bk158	bk159
bk130	bk160	bk161
bk131	bk162	bk163
bk132	bk164	bk165
bk133	bk166	bk167
bk134	bk168	bk169
bk135	bk170	bk171
bk136	bk172	bk173
bk137	bk174	bk175
bk138	bk176	bk177
bk139	bk178	bk179
bk140	bk180	bk181
bk141	bk182	bk183
bk142	bk184	bk185
bk143	bk186	bk187
bk144	bk188	bk189
bk145	bk190	bk191
bk146	bk192	bk193
bk147	bk194	bk195
bk148	bk196	bk197
bk149	bk198	bk199
bk150	bk200	bk201
bk151	bk202	bk203
bk152	bk204	bk205
bk153	bk206	bk207
bk154	bk208	bk209
bk155	bk210	bk211
bk156	bk212	bk213
bk157	bk214	bk215
bk158	bk216	bk217
bk159	bk218	bk219
bk160	bk220	bk221
bk161	bk222	bk223
bk162	bk224	bk225
bk163	bk226	bk227
bk164	bk228	bk229
bk165	bk230	bk231
bk166	bk232	bk233
bk167	bk234	bk235
bk168	bk236	bk237
bk169	bk238	bk239
bk170	bk240	bk241
bk171	bk242	bk243
bk172	bk244	bk245
bk173	bk246	bk247
bk174	bk248	bk249
bk175	bk250	bk251
bk176	bk252	bk253
bk177	bk254	bk255
bk178	bk256	bk257
bk179	bk258	bk259
bk180	bk260	bk261
bk181	bk262	bk263
bk182	bk264	bk265
bk183	bk266	bk267
bk184	bk268	bk269
bk185	bk270	bk271
bk186	bk272	bk273
bk187	bk274	bk275
bk188	bk276	bk277
bk189	bk278	bk279
bk190	bk280	bk281
bk191	bk282	bk283
bk192	bk284	bk285
bk193	bk286	bk287
bk194	bk288	bk289
bk195	bk290	bk291
bk196	bk292	bk293
bk197	bk294	bk295
bk198	bk296	bk297
bk199	bk298	bk299
bk200	bk300	bk301
bk302	bk303	bk304
bk305	bk306	bk307
bk308	bk309	bk310
bk311	bk312	bk313
bk314	bk315	bk316
bk317	bk318	bk319
bk320	bk321	bk322
bk323	bk324	bk325
bk326	bk327	bk328
bk329	bk330	bk331
bk332	bk333	bk334
bk335	bk336	bk337
bk338	bk339	bk340
bk341	bk342	bk343
bk344	bk345	bk346
bk347	bk348	bk349
bk350	bk351	bk352
bk353	bk354	bk355
bk356	bk357	bk358
bk359	bk360	bk361
bk362	bk363	bk364
bk365	bk366	bk367
bk368	bk369	bk370
bk371	bk372	bk373
bk374	bk375	bk376
bk377	bk378	bk379
bk380	bk381	bk382
bk383	bk384	bk385
bk386	bk387	bk388
bk389	bk390	bk391
bk392	bk393	bk394
bk395	bk396	bk397
bk398	bk399	bk400
bk401	bk402	bk403
bk404	bk405	bk406
bk407	bk408	bk409
bk410	bk411	bk412
bk413	bk414	bk415
bk416	bk417	bk418
bk419	bk420	bk421
bk422	bk423	bk424
bk425	bk426	bk427
bk428	bk429	bk430
bk431	bk432	bk433
bk434	bk435	bk436
bk437	bk438	bk439
bk440	bk441	bk442
bk443	bk444	bk445
bk446	bk447	bk448
bk449	bk450	bk451
bk452	bk453	bk454
bk455	bk456	bk457
bk458	bk459	bk460
bk461	bk462	bk463
bk464	bk465	bk466
bk467	bk468	bk469
bk470	bk471	bk472
bk473	bk474	bk475
bk476	bk477	bk478
bk479	bk480	bk481
bk482	bk483	bk484
bk485	bk486	bk487
bk488	bk489	bk490
bk491	bk492	bk493
bk494	bk495	bk496
bk497	bk498	bk499
bk500	bk501	bk502
bk503	bk504	bk505
bk506	bk507	bk508
bk509	bk510	bk511
bk512	bk513	bk514
bk515	bk516	bk517
bk518	bk519	bk520
bk521	bk522	bk523
bk524	bk525	bk526
bk527	bk528	bk529
bk530	bk531	bk532
bk533	bk534	bk535
bk536	bk537	bk538
bk539	bk540	bk541
bk542	bk543	bk544
bk545	bk546	bk547
bk548	bk549	bk550
bk551	bk552	bk553
bk554	bk555	bk556
bk557	bk558	bk559
bk560	bk561	bk562
bk563	bk564	bk565
bk566	bk567	bk568
bk569	bk570	bk571
bk572	bk573	bk574
bk575	bk576	bk577
bk578	bk579	bk580
bk581	bk582	bk583
bk584	bk585	bk586
bk587	bk588	bk589
bk590	bk591	bk592
bk593	bk594	bk595
bk596	bk597	bk598
bk599	bk600	bk601
bk602	bk603	bk604
bk605	bk606	bk607
bk608	bk609	bk610
bk611	bk612	bk613
bk614	bk615	bk616
bk617	bk618	bk619
bk620	bk621	bk622
bk623	bk624	bk625
bk626	bk627	bk628
bk629	bk630	bk631
bk632	bk633	bk634
bk635	bk636	bk637
bk638	bk639	bk640
bk641	bk642	bk643
bk644	bk645	bk646
bk647	bk648	bk649
bk650	bk651	bk652
bk653	bk654	bk655
bk656	bk657	bk658
bk659	bk660	bk661
bk662	bk663	bk664
bk665	bk666	bk667
bk668	bk669	bk670
bk671	bk672	bk673
bk674	bk675	bk676
bk677	bk678	bk679
bk680	bk681	bk682
bk683	bk684	bk685
bk686	bk687	bk688
bk689	bk690	bk691
bk692	bk693	bk694
bk695	bk696	bk697
bk698	bk699	bk700
bk701	bk702	bk703
bk704	bk705	bk706
bk707	bk708	bk709
bk710	bk711	bk712
bk713	bk714	bk715
bk716	bk717	bk718
bk719	bk720	bk721
bk722	bk723	bk724
bk725	bk726	bk727
bk728	bk729	bk730
bk731	bk732	bk733
bk734	bk735	bk736
bk737	bk738	bk739
bk740	bk741	bk742
bk743	bk744	bk745
bk746	bk747	bk748
bk749	bk750	bk751
bk752	bk753	bk754
bk755	bk756	bk757
bk758	bk759	bk760
bk761	bk762	bk763
bk764	bk765	bk766
bk767	bk768	bk769
bk770	bk771	bk772
bk773	bk774	bk775
bk776	bk777	bk778
bk779	bk780	bk781
bk782	bk783	bk784
bk785	bk786	bk787
bk788	bk789	bk790
bk791	bk792	bk793
bk794	bk795	bk796
bk797	bk798	bk799
bk800	bk801	bk802
bk803	bk804	bk805
bk806	bk807	bk808
bk809	bk810	bk811
bk812	bk813	bk814
bk815	bk816	bk817
bk818	bk819	bk820
bk821	bk822	bk823
bk824	bk825	bk826
bk827	bk828	bk829
bk830	bk831	bk832
bk833	bk834	bk835
bk836	bk837	bk838
bk839	bk840	bk841
bk842	bk843	bk844
bk845	bk846	bk847
bk848	bk849	bk850
bk851	bk852	bk853
bk854	bk855	bk856
bk857	bk858	bk859
bk860	bk861	bk862
bk863	bk864	bk865
bk866	bk867	bk868
bk869	bk870	bk871
bk872	bk873	bk874
bk875	bk876	bk877
bk878	bk879	bk880
bk881	bk882	bk883
bk884	bk885	bk886
bk887	bk888	bk889
bk890	bk891	bk892
bk893	bk894	bk895
bk896	bk897	bk898
bk899	bk900	bk901
bk902	bk903	bk904
bk905	bk906	bk907
bk908	bk909	bk910
bk911	bk912	bk913
bk914	bk915	bk916
bk917	bk918	bk919
bk920	bk921	bk922
bk923	bk924	bk925
bk926	bk927	bk928
bk929	bk930	bk931
bk932	bk933	bk934
bk935	bk936	bk937
bk938	bk939	bk940
bk941	bk942	bk943
bk944	bk945	bk946
bk947	bk948	bk949
bk950	bk951	bk952
bk953	bk954	bk955
bk956	bk957	bk958
bk959	bk960	bk961
bk962	bk963	bk964
bk965	bk966	bk967
bk968	bk969	bk970
bk971	bk972	bk973
bk974	bk975	bk976
bk977	bk978	bk979
bk980	bk981	bk982
bk983	bk984	bk985
bk986	bk987	bk988
bk989	bk990	bk991
bk992	bk993	bk994
bk995	bk996	bk997
bk998	bk999	bk1000

3. [4 Pts.] Write an XPath expression to find the title, price and the description in text of the last book in the catalog.

```
db:open("Section_A_XML_Data","Section_A_XML_Data.xml")
```

```
//book[12]/(title|price|description)/text()
```



4. [4 Pts.] Write an XPath expression to find the authors and titles of the books which cost more than 40 dollars, along with the respective prices.

```
db:open("Section_A_XML_Data","Section_A_XML_Data.xml")
```

```
//book[price>40]//(author|title)/text()
```

The screenshot shows a software interface with two main panels. The left panel, titled 'Result', displays the output of an XPath query. It shows '4 Results, 96 b' and lists the following text: 'Gambardella, Matthew', 'XML Developer's Guide', 'Galos, Mike', and 'Visual Studio 7: A Comprehensive Guide'. The right panel shows the XPath query being executed: `db:open("Section_A_XML_Data","Section_A_XML_Data.xml")` on line 1, and `//book[price>40]//(author|title)/text()` on line 2. The interface also includes a toolbar with various icons and a status bar at the bottom.

5. [4 Pts.] Write an XPath expression to find the authors and prices of the books belonging to Computer genre.

```
db:open("Section_A_XML_Data","Section_A_XML_Data.xml")
```

```
//book[genre="Computer"]/(author|price)/text()
```

The screenshot shows a software interface with two main panels. The left panel, titled 'Result', displays the output of an XPath query. It shows 8 results, totaling 89 bytes. The results are listed as follows:

Author	Price
Gambardella, Matthew	44.95
O'Brien, Tim	36.95
O'Brien, Tim	36.95
Galos, Mike	49.95

The right panel, titled 'Editor', shows the context and the XPath expression used for the query. The context is 'db:open("Section_A_XML_Data","Section_A_XML_Data.xml")'. The XPath expression is:

```
1 db:open("Section_A_XML_Data","Section_A_XML_Data.xml")
2 //book[genre="Computer"]/(author|price)/text()
3
```

Below the editor, there is a diagram showing the XML structure, with a tree view of the document. The diagram shows a root node with several child nodes, and the selected node is highlighted in blue.

6. [6 Pts.] Write an XQuery (FLWOR) script to find the titles of the books arranged in ascending order of price, of which the price are more than 30 dollars.

```
for $bk in db:open('Section_A_XML_Data')//book
let $title:=$bk/title
let $price:=$bk/price
where $price>30
order by $price
return $title/text()
```

The screenshot displays a software interface with two main panels. The left panel, titled 'Result', shows the output of an XQuery: '4 Results, 131 b' followed by a list of book titles: 'Microsoft .NET: The Programming Bible', 'MSXML3: A Comprehensive Guide', 'XML Developer's Guide', and 'Visual Studio 7: A Comprehensive Guide'. The right panel contains an XQuery script in a text editor. The script is as follows:

```
1 for $bk in db:open('Section_A_XML_Data')//book
2 let $title:=$bk/title
3 let $price:=$bk/price
4 where $price>30
5 order by $price
6 return $title/text()
```

Below the script, there is a status bar with a green checkmark and the text 'OK'. At the bottom of the right panel, there is a diagram showing a hierarchical structure with a root node labeled 'Section_A_XML_Data.xml' and several child nodes below it.

7. [6 Pts.] Write an XQuery (FLWOR) script to provide only the descriptions of the books, which cost less than 5 dollars.

```
for $bk in db:open('Section_A_XML_Data')//book
let $description:=$bk/description
let $price:=$bk/price
where $price<5
return $description
```

The screenshot shows a software interface with two main panels. The left panel, titled 'Result', displays the output of an XQuery script. It shows three XML elements, each starting with a blue-colored opening tag <description>. The first element describes a meeting between Carla and Paul at an ornithology conference. The second describes a deep sea diver finding true love twenty thousand leagues beneath the sea. The third describes an anthology of horror stories about roaches, centipedes, scorpions, and other insects. The right panel is a script editor with a toolbar and a code area. The code area contains the XQuery script from the previous block, with line numbers 1 through 8 on the left. The script is:
1 for \$bk in db:open('Section_A_XML_Data')//book
2 let \$description:=\$bk/description
3 let \$price:=\$bk/price
4 where \$price<5
5 return \$description
6
7
8
Below the code area is a status bar with a green checkmark icon and the text 'OK'. At the bottom of the interface is a diagram showing a hierarchical tree structure with a root node, three intermediate nodes, and a large number of leaf nodes represented by small blue rectangles.

SessionDesktop

Find Find...

3 Results, 356 b

Result

```
<description>When Carla meets Paul at an
    ornithology
    conference, tempers fly as
feathers get ruffled.</description>
<description>A deep sea diver finds true
    love twenty
    thousand leagues beneath the sea.<
/description>
<description>An anthology of horror
stories about roaches,
    centipedes, scorpions and other
insects.</description>
```

file*

```
1 for $bk in db:open('Section_A_XML_Data')//book
2 let $description:=$bk/description
3 let $price:=$bk/price
4 where $price<5
5 return $description
6
7
8
```

Context: db:open("Section_A_XML_Data")

OK

8. 8. [6 Pts.] Write an XQuery (FLWOR) script which gives the various genre along with the text of the title of the books in them.

```
for $bk in db:open('Section_A_XML_Data')//book
```

```
let $title:=$bk/title
```

```
let $genre:=$bk/genre
```

```
order by $genre
```

```
return $genre|$title/text()
```

The screenshot shows a software interface with two main panes. The left pane, titled 'Result', displays 24 results with a total size of 558 bytes. It lists book titles and their genres, with XML tags for the genre. The right pane, titled 'Editor', shows the XQuery script being executed. The script is a FLWOR expression that iterates over books, extracts their titles and genres, orders them by genre, and returns the genre and title text. The interface includes a 'Find' bar at the top, a toolbar with icons for file operations and execution, and a status bar at the bottom showing 'OK' and '5 : 9'.

SessionDesktop

Find Find...

24 Results, 558 b Result

XML Developer's Guide
<genre>Computer</genre>
Microsoft .NET: The
Programming Bible
<genre>Computer</genre>
MSXML3: A Comprehensive Guide
<genre>Computer</genre>
Visual Studio 7: A
Comprehensive Guide
<genre>Computer</genre>
Midnight Rain
<genre>Fantasy</genre>
Maevae Ascendant
<genre>Fantasy</genre>
Oberon's Legacy
<genre>Fantasy</genre>
The Sundered Grail
<genre>Fantasy</genre>
Creepv Crawlies

File*

```
1 for $bk in db:open('Section_A_XML_Data')//book
2 let $title:=$bk/title
3 let $genre:=$bk/genre
4 order by $genre
5 return $genre|$title/text()
6
7
8
9
```

Context: db:open("Section_A_XML_Data") Editor

OK 5 : 9

9. [6 Pts.] Write an XQuery (FLWOR) script which gives the description text showing that the book belongs to Fantasy genre.

```
for $bk in db:open('Section_A_XML_Data')//book
let $description:=$bk/description
let $genre:=$bk/genre
where $genre="Fantasy"
order by $genre
return $description/text()
```

The screenshot shows a BaseX XQuery editor interface. The left pane displays the query results, which are 4 XML documents, totaling 575 bytes. The right pane shows the XQuery script being executed. The script is a FLWOR expression that iterates over all book elements in the 'Section_A_XML_Data' database, filters for books with the genre 'Fantasy', orders them by genre, and returns the text of their descriptions. The results pane shows the first result, which is an XML document with a single text node containing the description of a book. The script pane shows the following code:

```
1 for $bk in db:open('Section_A_XML_Data')//
2 let $description:=$bk/description
3 let $genre:=$bk/genre
4 where $genre="Fantasy"
5 order by $genre
6 return $description/text()
```

The results pane shows the following XML document:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" >
  <text>
    A former architect battles corporate
    zombies,
    an evil sorceress, and her own
    childhood to become queen
    of the world.
    After the collapse of a nanotechnology
    society in England, the young
    survivors lay the
    foundation for a new society.
    In post-apocalypse England, the mysterious
    agent known only as Oberon helps to
    create a new life
    for the inhabitants of London. Sequel
    to Maeve
    Ascendant.
    The two daughters of Maeve, half-sisters,
    battle one another for control of
    England. Sequel to
    Oberon's Legacy.
  </text>
</xml>
```

10. [6 Pts.] Write an XQuery (FLWOR) script which gives the list of authors whose books cost less than 30 dollars and provides the titles of the books otherwise.

```
for $bk in db:open('Section_A_XML_Data')//book
```

```
let $print:=if($bk/price<30) then $bk/author
```

```
else $bk/title
```

```
return $print/text()
```

The screenshot shows a software interface with a 'Find' bar at the top left. Below it, a list of 12 results is displayed, showing authors and book titles. To the right, a 'Result' pane shows the XQuery script being executed. Below the script, there is a search bar with the text 'ore/book[price>30]/tit' and a 'Replace with...' field. At the bottom, there is a diagram showing a hierarchical structure of nodes.

Find Find... 12 Results, 245 b

XML Developer's Guide
Ralls, Kim
Corets, Eva
Corets, Eva
Corets, Eva
Randall, Cynthia
Thurman, Paula
Knorr, Stefan
Kress, Peter
Microsoft .NET: The Programming Bible
MSXML3: A Comprehensive Guide
Visual Studio 7: A Comprehensive Guide

Result

Context: db:open("Se")

```
1 for $bk in db:open('Section_A_XML_Data')//
2 let $print:=if($bk/price<30) then $bk/aut
3     else $bk/title
4 return $print/text()
```

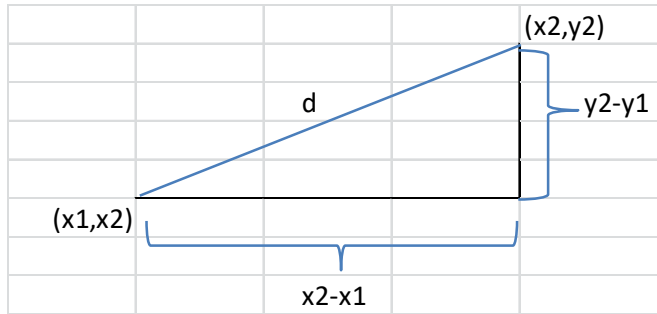
ore/book[price>30]/tit Replace with...

OK

Section B

1. [35 Pts.] Use the K-means algorithm to cluster this dataset. You can initiate the calculation by assuming $K=2$ and assume that the records with RIDs 103, and 104 are used as the initial cluster centroids.

Euclidean distance $d = ((x_2 - x_1)^2 + (y_2 - y_1)^2)^{.5}$



1st iteration

$K=2$, c_1 = RID 103, c_2 = RID 104

RID	x_i		c_1		c_2		Distance 1 ($ x_i - c_1 $)	Distance 2 ($ x_i - c_2 $)	Nearest Cluster	New Centroid		
	age	yrs	age	yrs	age	yrs					age	yrs
101	30	5	50	15	25	5	22.4	5.0	c_2	c_1	51.7	21.7
102	50	25	50	15	25	5	10.0	32.0	c_1	c_2	28.3	6.7
103	50	15	50	15	25	5	0.0	26.9	c_1			
104	25	5	50	15	25	5	26.9	0.0	c_2			
105	30	10	50	15	25	5	20.6	7.1	c_2			
106	55	25	50	15	25	5	11.2	36.1	c_1			
							lowest distance					

c_1 = mean of RID 102,103,106

$(50+50+55)/3 = 51.7$

New centroids $c_1 = (51.7, 21.7)$, $c_2 = (28.3, 6.7)$

2nd iteration

$c_1 = (51.7, 21.7)$, $c_2 = (28.3, 6.7)$

RID	x_i		c_1		c_2		Distance 1 ($ x_i - c_1 $)	Distance 2 ($ x_i - c_2 $)	Nearest Cluster	New Centroid		
	age	yrs	age	yrs	age	yrs					age	yrs
101	30	5	51.7	21.7	28.3	6.7	27.3	1.7	c_2	c_1	51.7	21.7
102	50	25	51.7	21.7	28.3	6.7	3.7	29.5	c_1	c_2	28.3	6.7
103	50	15	51.7	21.7	28.3	6.7	6.9	23.9	c_1			
104	25	5	51.7	21.7	28.3	6.7	31.4	3.3	c_2			
105	30	10	51.7	21.7	28.3	6.7	24.6	5.3	c_2			
106	55	25	51.7	21.7	28.3	6.7	4.7	33.3	c_1			

New centroids $c_1 = (51.7, 21.7)$, $c_2 = (28.3, 6.7)$

2. [15 Pts.] Provide a brief description on the difference between describing discovered knowledge using clustering and describing it using classification?

There is no difference in distance using iteration 1 and 2. We identified 2 groups using clustering. So the 2 groups will be RID 102,103,106 and RID 101,104,105.

In Classification method, we classify sample in to known groups. We may use our knowledge about data in to groups . In this example we can classify using age range. Age between 25 and 49 , age between 50 and 55.

In classification we use our knowledge to put sample in to classes. In clustering we do it randomly without knowledge.