XML Query Use Cases in X-DEVICE

Queries		**X-DEVICE**
Use Case "XMP"		
Q1	List books published by Addison Wesley after 1991, including their year and title.	`if B@book(title:T,publisher='Addison Wesley',year:Y>1991)` `then xml_result(bib(book1(title:T,year:Y)))`
Q2	Create a flat list of all the title-author pairs, with each pair enclosed in a "result" element.	`if b@book(title:T,author ∋ A)` `then xml_result(results(result(title:T,author:A)))`
Q3	For each book in the bibliography, list the title and authors, grouped inside a "result" element	`if b@book(title:T,author:A)` `then xml_result(results(result(title:T,author:A)))`
Q4	For each author in the bibliography, list the author's name and the titles of all books by that author, grouped inside a "result" element	`if A@author and` `   B@book(title:T,author ∋ A)` `then xml_result(results(result(author:A,title:list(T))))`
Q5	For each book found at both `bn.com` and `amazon.com`, list the title of the book and its price from each source	`if B@book(title:T,price:P1) and` `   E@entry(title:T,price:P2)` `then xml_result(bs_w_prc(b_w_prc(title:T,prc_amazon:P2,prc_bn:P1)))`
Q6	For each book, list the title and first two authors, and an empty "et-al" element if the book has additional authors	`if B@book(title:T,author` $\ni_{=<2}$ `A)` `then xml_result(bib(book1(title:T,author:list(A))))` `if B1@book1(title:T) and` `   B@book(title:T,author:A) and` `   prolog{length(A,L), L>2}` `then B1@book1(∅et_al)` *or* `if B@book(title:T,author:A) and` `   prolog{length(A,L), L>2, first_n(2,A,A1)}` `then xml_result(bib(book1(title:T,author:A1,∅et_al:yes)))` `if B@book(title:T,author:A) and` `   prolog{length(A,L), L=<2}` `then xml_result(bib(book1(title:T,author:A,∅et_al:no)))` *or just* `book1(title:T,author:A,∅et_al:no)`
Q7	List the titles and years of all books published by Addison Wesley after 1991, in alphabetic order	`if B@book(title:T,publisher='Addison Wesley',year:Y>1991)` `then xml_sorted([T],bib(book1(title:T,year:Y)))`

Queries		X-DEVICE
Q8	Find books in which some element has a tag ending in "or" and the same element contains the string "Suciu" (at any level of nesting). For each such book, return the title and the qualifying element	`if B@book(title:T,Att:Val =* 'Suciu') and prolog{suffix(Att,or)} then xml_result(book1(title:T,Att:Val))`
Q8a	Find books in which the 'author' element contains the string "Suciu" (at any level of nesting). For each such book, return the title and the author element	`if B@book(title:T,author:Val =* 'Suciu') then xml_result(book1(title:T,author:Val))`
Q9	In the document "`books.xml`", find all section or chapter titles that contain the word "XML", regardless of the level of nesting	`if D@xml_doc(uri='books.xml',title.*.root_elem:T $ 'XML') then xml_result(results(title:list(T)))`
Q10	In the document "`prices.xml`", find the minimum price for each book, in the form of a "minprice" element with the book title as its title attribute	`if D@xml_doc(uri='prices.xml',title.book.root_elem:T,price.book.root_elem:P) then xml_result(results(minprice(^title:T,:min(P))))`
Q11	From each book with an author, return the book with its title and authors. For each book with an editor, return a reference with the book title and the editor's affiliation	`if B@book(title:T,author:A) then xml_result(book1(title:T,author:A))` `if B@book(title:T,affiliation.editor:A) then xml_result(reference(title:T,org:A))`
Q12	Find pairs of books that have different titles but the same **set** of authors	`if B1@book(title:T1,author:A1) and B2@book(title:T2\=T1,author:A2) and prolog{same_set(A1,A2)} then xml_result(bib(book_pair(title:[T1,T2])))`
Use Case "TREE"		
Q1	Prepare a (nested) table of contents for Book1, listing all the sections and their titles. Preserve the original attributes of each <section> element, if any	`if B@book(title='Book1',section* ∍ S) and S@section(title:T) then section1(title:T,!org_obj:S)` `if S1@section1(!org_obj:S) and S@section(section ∍ CS) and CS1@section1(!org_obj=CS) then S1@section1(section1:list(CS1))` `if S1@section1(!org_obj:S) and S@section(^Att:Val) then S1@section1(^Att:Val)` `if B@book(title='Book1',section ∍ S) and S1@section1(!org_obj:S) then xml_result(toc(section1:list(S1)))`

Queries		**X-DEVICE**

Q2	Prepare a (flat) figure list for Book1, listing all the figures and their titles. Preserve the original attributes of each <figure> element, if any	if B@book(title='Book1', figure.section* ∋ F) and
		   F@figure(title:T)
		then xml_result(figurelist(figure1(title:T,!org_obj:F)))
		
		if F1@figure1(!org_obj:F) and
		   F@figure(^Att:Val)
		then F1@figure1(^Att:Val)

Q3	How many sections are in Book1, and how many figures?	if B@book(title='Book1',section* ∋ S)
		then xml_result(section_count:count(S))
		
		if B@book(title='Book1', figure.section* ∋ F)
		then xml_result(figure_count:count(F))

Q4	How many top-level sections are in Book1?	if B@book(title='Book1',section ∋ S) then xml_result(top_section_count:count(S))

Q5	Make a flat list of the section elements in Book1. In place of its original attributes, each section element should have two attributes, containing the title of the section and the number of figures immediately contained in the section	if B@book(title='Book1',section* ∋ S) and
		   S@section(title:T,figure:F) and
		   prolog{length(F,FC)}
		then xml_result(section_list(section1(^title:T,^figcount:FC)))

Q6	Make a nested list of the section elements in Book1, preserving their original attributes and hierarchy. Inside each section element, include the title of the section and an element that includes the number of figures immediately contained in the section	if B@book(title='Book1',section* ∋ S) and
		   S@section(title:T,figure:F) and
		   prolog{length(F,FC)}
		then section1(title:T,figcount:FC,!org_obj:S)
		
		if S1@section1(!org_obj:S) and
		   S@section(section ∋ CS) and
		   CS1@section1(!org_obj=CS)
		then S1@section1(section1:list(CS1))
		
		if S@section(^Att:Val) and
		   S1@section1(!org_obj:S)
		then S1@section1(^Att:Val)
		
		if B@book(title='Book1',section ∋ S) and
		   S1@section1(!org_obj:S)
		then xml_result(section_summary(section1:list(S1)))

Use Case "SEQ"		

Q1	In the Procedure section of Report1, what Instruments were used in the second Incision?	if Report1@report(section:S) and
		   S@section(section_title='Procedure',incision.section_content $\ni_2$ I) and
		   I@incision(instrument:Instr)
		then xml_result(result(instrument:Instr))

Queries		**X-DEVICE**
Q2	In the Procedure section of Report1, what are the first two Instruments to be used?	if Report1@report(section:S) and S@section(section_title='Procedure',instrument.C.section_content $\ni_{=<2}$ I) then xml_result(result(instrument:list(I)))
Q2a	In the Procedure section of Report1, what are the first two Instruments to be used? (Depth of instrument tag is unknown)	if R@report(section_title.section='Procedure',instrument.*.section_content.section $\ni_{=<2}$ I) then xml_result(instrument:list(I))
Q3	In Report1, what Instruments were used in the first two Actions after the second Incision?	if Report1@report(section:S) and S@section(incision.section_content $\ni_{2}$ I) and S@section(action.section_content $\ni_{\{after(I),\ =<2\}}$ A) and A@action(instrument $\ni$ I2) then xml_result(result(instrument:list(I2)))
Q4	In Report1, find "Procedure" sections where no Anesthesia element occurs before the first Incision	if Report1@report(section:S) and S@section(section_title='Procedure', incision.section_content $\ni_{1}$ I) and not S@section(anesthesia.section_content $\ni_{before(I)}$ A) then xml_result(result(section:list(S)))
Q5	In Report1, what happened between the first Incision and the second Incision?	if Report1@report(section:S) and S@section(incision.section_content $\ni_{<=2}$ [I1,I2]) and S@section(H.section_content $\ni_{between(I1,I2)}$ W) then xml_result(result(happened:list(W)))
Use Case "R"		
Q1	List the item number and description of all bicycles that currently have an auction in progress, ordered by item number	if I@item_tuple(itemno:N,description:D$'bicycle',start_date=<today(),end_date>=today()) then xml_sorted([N],result(item_tuple(itemno:N,description:D)))
Q2	For all bicycles, list the item number, description, and highest bid (if any), ordered by item number.	if I@item_tuple(itemno:N,description:D$'bicycle') and B@bid_tuple(itemno:N,bid:BP) then xml_sorted([N],result(item_tuple(itemno:N,description:D,highest_bid:max(BP))))
Q3	Find cases where a user with a rating worse (alphabetically, greater) than "C" is offering an item with a reserve price of more than 1000	if U@user_tuple(userid:UID,name:UN,rating:R@>'C') and I@item_tuple(description:D,offered_by:UID,reserve_price:RP>1000) then xml_result(result(warning(user_name:UN,user_rating:R, item_description:D,reserve_price:RP)))
Q4	List item numbers and descriptions of items that have no bids	if I@item_tuple(itemno:N,description:D) and not B@bid_tuple(itemno:N) then xml_result(result(no_bid_item(itemno:N,description:D)))
Q5	For bicycle(s) offered by Tom Jones, list the item number, description, highest bid (if any), and name of the highest bidder, ordered by item number.	if U@user_tuple(userid:UID,name='Tom Jones') and I@item_tuple(itemno:N,description:D$'bicycle',offered_by:UID) and B@bid_tuple(itemno:N,bid:BP) and then xml_sorted([N],result(auction_item(itemno:N,description:D,highest_bid:max(BP)))) if A@auction_item(itemno:N,highest_bid:HB) and B@bid_tuple(itemno:N,bid:HB,userid:BUID) U@user_tuple(userid:BUID,name:BN) then A@auction_item(bidder:BN)

Queries		X-DEVICE
Q6	For each item whose highest bid is more than twice its reserve price, list the item number, description, reserve price, and highest bid.	`if B@bid_tuple(itemno:N,bid:BP)` `then higest_bid(itemno:N,high_bid:max(BP))` `if H@higest_bid(itemno:N,high_bid:HB) and` `I@item_tuple(itemno:N,description:D,reserve_price:P<HB/2)` `then xml_result(successful_item(itemno:N,description:D,reserve_price:P,high_bid:HB))`
Q7	Find the highest bid ever made for a bicycle or tricycle.	`if I@item_tuple(itemno:N,description$'bicycle';$'tricycle') and` `B@bid_tuple(itemno:N,bid:BP)` `then xml_result(high_bid(bid:max(BP)))`
Q8	How many items were actioned (auction ended) in March 1999?	`if I@item_tuple(end_date>="1999-03-01"&=<"1999-03-31")` `then xml_result(result(item_count:count(I)))`
Q9	List the number of items auctioned each month in 1999 for which data is available, ordered by month.	`if I@item(end_date $m M)` `then xml_sorted([M],result(monthly_result(month:M,item_count:count(I))))`
Q10	For each item that has received a bid, list the item number, the highest bid, and the name of the highest bidder, ordered by item number	`if B@bid_tuple(itemno:N,bid:Bid)` `then xml_sorted([N],result(high_bid(itemno:N,bid:max(Bid))))` `if H@high_bid(itemno:N,bid:MB) and` `B@bid_tuple(itemno:N,bid:MB,userid:BUID)` `U@user_tuple(userid:BUID,name:BN)` `then H@high_bid(bidder:BN)`
Q11	List the item number and description of the item(s) that received the highest bid ever recorded, and the amount of that bid.	`if B@bid_tuple(bid:Bid)` `then high_bid(bid:max(Bid))` `if H@high_bid(bid:HB) and` `B@bid_tuple(itemno:N,bid=HB) and` `I@item_tuple(itemno:N,description:D)` `then xml_result(result(expensive_item(itemno:N,description:D,high_bid:H)))`
Q12	List the item number and description of the item(s) that received the largest number of bids, and the number of bids it (or they) received.	`if B@bid_tuple(itemno:N,bid:Bid)` `then bid_count(itemno:N,bids:count(B))` `if B@bid_count(itemno:N,bids:BC) and` `not B1@bid_count(bids>BC) and` `I@item_tuple(itemno:N,description:D)` `then xml_result(result(popular_item(itemno:N,description:D,bid_count:BC)))`
Q13	For each user who has placed a bid, give the userid, name, number of bids, and average bid, in order by userid	`if B@bid_tuple(userid:UID,bid:Bid) and` `U@user_tuple(userid:UID,name:Name)` `then xml_sorted([UID],` `result(bidder(userid:UID,name:Name,bidcount:count(Bid),avgbid:avg(Bid))))`
Q14	List item numbers and average bids for items that have received three or more bids, in descending order by average bid	`if B@bid_tuple(itemno:N,bid:Bid)` `then bid_count(itemno:N,bids:count(B),avgbid:avg(B))` `if B@bid_count(itemno:N,bids>=3,avgbid:Avg)` `then xml_sorted([Avg],result(popular_item(itemno:N,avgbid:Avg)))`

Queries		X-DEVICE
Q15	List names of users who have placed multiple bids of at least $100 each.	`if B@bid_tuple(userid:UID,bid>=100)`
		`then bid_count(userid:UID,bids:count(B))`
		`if B@bid_count(userid:UID,bids>1) and`
		`   U@user_tuple(userid=UID,name:Name)`
		`then xml_result(result(big_spender:list(Name)))`
Q16	List all registered users in order by userid; for each user, include the userid, name, and an indication of whether the user is active (has at least one bid on record) or inactive (has no bid on record).	`if U@user_tuple(userid:UID,name:Name) and`
		`   B@bid_tuple(userid=UID)`
		`then xml_sorted([UID],result(user(userid:UID,name:Name,status:active)))`
		`if U@user_tuple(userid:UID,name:Name) and`
		`   not B@bid_tuple(userid=UID)`
		`then user(userid:UID,name:Name,status:inactive)`
Q17	List the names of users, if any, who have bid on every item.	`if U@user_tuple(userid:UID,name:Name) and`
		`   not X@does_not_have_some_bid(userid:UID)`
		`then xml_result(frequent_bidder(name:Name))`
		`if I@item_tuple(itemno:N) and`
		`   U@user_tuple(userid:UID) and`
		`   not B@bid_tuple(itemno=N,userid=UID)`
		`then does_not_have_some_bid(userid:UID)`
Q18	List all users in alphabetic order by name. For each user, include descriptions of all the items (if any) that were bid on by that user, in alphabetic order.	`if U@user_tuple(userid:UID,name:Name) and`
		`   B@bid_tuple(userid=UID,itemno:N) and`
		`   I@item_tuple(itemno=N,description:D)`
		`then xml_sorted([Name],result(user(name:Name,bid_on_item:ord_list(D))))`
Use Case "SGML"		
Q1	Locate all paragraphs in the report (all "para" elements occurring anywhere within the "report" element).	`if R@report(para.* ∋ P)`
		`then xml_result(result(para:list(P))`
Q2	Locate all paragraph elements in an introduction (all "para" elements directly contained within an "intro" element).	`if R@report(para.intro.* ∋ P)`
		`then xml_result(result(para:list(P))`
Q3	Locate all paragraphs in the introduction of a section that is in a chapter that has no introduction (all "para" elements directly contained within an "intro" element directly contained in a "section" element directly contained in a "chapter" element. The "chapter" element must not directly contain an "intro" element).	`if R@report(intro.chapter \∋ I, para.intro.section.chapter ∋ P)`
		`then xml_result(result(para:list(P))`

Queries		X-DEVICE
Q4	Locate the second paragraph in the third section in the second chapter (the second "para" element occurring in the third "section" element occurring in the second "chapter" element occurring in the "report").	if R@report(para$_2$.section$_3$.chapter$_2$:P) then xml_result(result(para:P))
Q5	Locate all classified paragraphs (all "para" elements whose "security" attribute has the value "c").	`if P@para(^security='c')` `then xml_result(result(para:list(P))`
Q6	List the short titles of all sections (the values of the "shorttitle" attributes of all "section" elements, expressing each short title as the value of a new element.)	`if S@section(^shorttitle:ST)` `then xml_result(result(stitle:list(ST)))`
Q7	Locate the initial letter of the initial paragraph of all introductions (the first character in the content [character content as well as element content] of the first "para" element contained in an "intro" element).	if I@intro(*.para$_1$ ∋ C) and prolog{sub_string(C,1,C1)} then xml_result(result(first_letter:list(C1)))
Q8a	Locate all sections with a title that has "is SGML" in it (all "section" elements that contain a "title" element that has the consecutive characters "is SGML" in its content). The string can be interrupted by sub-elements.	`if S@section(title↑ $ 'is SGML')` `then xml_result(result(section:list(S)))`
Q8b	Same as (Q8a), but the string cannot be interrupted by sub-elements.	`if S@section(*.title $ 'is SGML')` `then xml_result(result(section:list(S)))`
Q9	Locate all the topics referenced by a cross-reference anywhere in the report (all the "topic" elements whose "topicid" attribute value is the same as an "xrefid" attribute value of any "xref" element).	`if T@topic(^topicid:ID) and` `X@xref(^xrefid=ID)` `then xml_result(result(topic:list(T)))`
Q10	Locate the closest title preceding the cross-reference ("xref") element whose "xrefid" attribute is "top4" (the "title" element that would be touched last before this "xref" element when touching each element in document order).	`if P@Element(title:T, ^xrefid.xref.+='top4')` `then xml_result(result(title:T))`
Use Case "TEXT"		
Q1	Find all news items where the name "Foobar Corporation" appears in the title.	`if N@news_item(title:T $ 'Foobar Corporation')` `then xml_result(result(title:list(T)))`

Queries		X-DEVICE
Q2	Find news items where the Foobar Corporation and one or more of its partners are mentioned in the same paragraph and/or title. List each news item by its title and date.	`if C@company(name='Foobar Corporation',partner.partners ∋ P) and` `N@news_item(title:T $ 'Foobar Corporation' & $ P,date:D)` `then xml_result(news_item1(title:T,date:D))` `if C@company(name='Foobar Corporation',partner.partners ∋ P) and` `N@news_item(title:T,*.par.content $ 'Foobar Corporation' & $ P,date:D)` `then xml_result(news_item1(title:T,date:D))`
Q3	Find titles of news items where Foobar Corporation and one or more of its partners are mentioned in the same sentence, but none of its competitors are mentioned in the news item. (The "." character designates the end of a sentence.)	`if C@company(name='Foobar Corporation',partner.partners ∋ P,competitor.competitors:CO) and` `N@news_item(*:Text) and` `prolog{contains_in_same_sentence(Text,'Foobar Corporation',P)}` `then tmp_elem1(tmp_val:T,tmp_obj1:N,tmp_obj2:CO)` `if XX1@tmp_elem1(tmp_val:T,tmp_obj1:N)` `not( XX1@tmp_elem1(tmp_obj2 ∋ CO) and` `N@news_item(* $ CO) )` `then xml_result(result(title:T))`
Q4	Find news items where a company and one of its partners is mentioned in the same news item and the news item is not authored by the company itself.	`if C@company(name:Name,partner.partners ∋ P) and` `N@news_item(* $ Name & $ P,author \= Name)` `then xml_result(result(news_item:list(N)))`
Q5	For each news item that is relevant to the Gorilla Corporation, create an "item summary" element. The content of the item summary is the content of the title, date, and first paragraph of the news item, separated by periods. A news item is relevant if the name of the company is mentioned anywhere within the content of the news item.	`if N@news_item(title:T,*.content $ 'Gorilla Corporation',date:D) and` `N@news_item(par.content ∋`$_1$` PAR)` `then xml_result(item_summary(title:T,date:D,par:PAR))`
Q6	Find news items where two company names and some form of the word "acquire" appear in the title or in the same sentence in one of the paragraphs. A company name is defined as the content of a <name>, <partner>, or <competitor> element within a <company> element.	`if C@company(name:Name)` `then company_names(name:Name)` `if C@company(partner.partners ∋ Name)` `then company_names(name:Name)` `if C@company(competitor.competitors ∋ Name)` `then company_names(name:Name)` `if C1@company_names(name:Name1) and` `C2@company_name(name:Name2\=Name1) and` `N@news_item(title:T) and` `prolog{contains_stems_in_same_sentence(T,Name1,Name2,'acquire')}` `then xml_result(result(news_item:list(N)))`

Queries		X-DEVICE

		`if C1@company_names(name:Name1) and`
		`   C2@company_name(name:Name2\=Name1) and`
		`   N@news_item(par.content ∋ PAR) and`
		`   prolog{contains_stems_in_same_sentence(PAR,Name1,Name2,'acquire')}`
		`then xml_result(result(news_item:list(N)))`

Use Case "PARTS"		

Q1	Convert the sample document from "partlist" format to "parttree" format (see DTD section for definitions). In the result document, part containment is represented by containment of one <part> element inside another. Each part that is not part of any other part should appear as a separate top-level element in the output document.	`if P@part(^partid:ID,^name:Name)`
		`then part1(^partid:ID,^name:Name)`
		
		`if PP1@part1(^partid:ID) and`
		`   P1@part(^partid:ID,^parent:Parent) and`
		`   PP2@part1(^partid:Parent)`
		`then PP2@part1(part:list(PP1))`
		
		`if P1@part1 and`
		`   not P2@part1(part ∋ P1)`
		`then xml_result(parttree(part1:list(P1)))`

Use Case "REF"		

Q1	Find Martha's spouse.	`if P@person(^name='Martha',^spouse:SP) and`
		`   P1@person(^name=SP)`
		`then shallow_result(result(person:P1))`

Q2	Find Joe's children.	`if P@person(^name='Joe',person ∋ Ch)`
		`then shallow_result(result(person:list(Ch)))`
		
		`if P@person(^name='Joe',^spouse:S) and`
		`   P1@person(^name=S,person ∋ Ch)`
		`then shallow_result(result(person:list(Ch)))`

Q3	Find parents of athletes.	`if P@person(^job.person='Athlete')`
		`then shallow_result(result(person:list(P)))`
		
		`if P1@person(^job.person='Athlete',^spouse:SP) and`
		`   P2@person(^name=SP)`
		`then shallow_result(result(person:list(P)))`

Q4	Find people who have the same job as one of their parents.	`if P@person(^job:Job) and`
		`   P1@person(person ∋ P,^job=Job)`
		`then shallow_result(person:P)`
		
		`if P@person(^job:Job) and`
		`   P1@person(person ∋ P,^spouse:SP) and`
		`   P2@person(^name=SP,^job=Job)`
		`then shallow_result(person:P)`

Queries		X-DEVICE
Q5	List names of parents and children who have the same job, and their jobs.	`if P@person(^name:Child,^job:Job) and`
		`   P1@person(^name:Parent,person ∋ P,^job=Job)`
		`then xml_result(result(match(^parent:Parent,^child:Child,^job:Job)))`
		`if P@person(^name:Child,^job:Job) and`
		`   P1@person(person ∋ P,^spouse:Parent) and`
		`   P2@person(^name=Parent,^job=Job)`
		`then xml_result(result(match(^parent:Parent,^child:Child,^job:Job)))`
Q6	Find Bill's grandchildren.	`if P@person(^name='Bil',person.person ∋ G)`
		`then shallow_result(result(person:list(G)))`
		`if P@person(^name='Bil',^spouse.person:SP) and`
		`   P1@person(^name=SP,person ∋ G)`
		`then shallow_result(result(person:list(G)))`
		`if P@person(^name='Bil',^spouse:SP) and`
		`   P1@person(^name=SP,person.person ∋ G)`
		`then shallow_result(result(person:list(G)))`
		`if P@person(^name='Bil',^spouse:SP) and`
		`   P1@person(^name=SP,^spouse.person:SP1) and`
		`   P2@person(^name=SP1,person ∋ G)`
		`then shallow_result(result(person:list(G)))`
Q7	List name-pairs of grandparents and grandchildren.	`if P@person(^name=GPName,^name.person.person:GCName)`
		`then xml_result(result(grandparent(^name:GPName,grandchild:GCName)))`
		`if P@person(^name=GPName,^spouse.person:SP) and`
		`   P1@person(^name=SP,^name.person:GCName)`
		`then xml_result(result(grandparent(^name:GPName,grandchild:GCName)))`
		`if P@person(^name=GPName,^spouse:SP) and`
		`   P1@person(^name=SP,^name.person.person:GCName)`
		`then xml_result(result(grandparent(^name:GPName,grandchild:GCName)))`
		`if P@person(^name=GPName,^spouse:SP) and`
		`   P1@person(^name=SP,^spouse.person:SP1) and`
		`   P2@person(^name=SP1,^name.person:GCName)`
		`then xml_result(result(grandparent(^name:GPName,grandchild:GCName)))`

Queries		X-DEVICE

Q8	Find Dave's parents-in-law (parents of his spouse, if any).	`if P@person(^name='Dave',^spouse:SP) and`
		`   P1@person(^name.person=SP)`
		`then shallow_result(result(person:list(P1)))`
		
		`if P@person(^name='Dave',^spouse:SP) and`
		`   P1@person(^name.person=SP,^spouse:SP1) and`
		`   P2@person(^name=SP1)`
		`then shallow_result(result(person:list(P2)))`

Q9	Find people with no children.	`if P@person(person=[])`
		`then shallow_result(result(person:list(P)))`

Q10	Find single parents (people with children but no spouse)	`if P@person(person\=[]) and`
		`   not P@person(^spouse:SP)`
		`then shallow_result(result(person:list(P)))`

Q11	List the names of all Joe's descendants. Show each descendant as an element with the descendant's name as content and his or her marital status and number of children as attributes. Sort the descendants in descending order by number of children, and secondarily in alphabetical order by name.	`if P@person(^name:Name,person ∋ D)`
		`then desc(^name:Name,descendant:D)`
		
		`if P@person(^name:Name,^spouse:SP) and`
		`   P1@person(^name=SP,person ∋ D)`
		`then desc(^name:Name,descendant:D)`
		
		`if D@desc(^name:Name,person.descendant ∋ D1)`
		`then desc(^name:Name,descendant:D1)`
		
		`if D@desc(^name:Name,^spouse.descendant:SP) and`
		`   P@person(^name=SP,person ∋ D1)`
		`then desc(^name:Name,descendant:D1)`
		
		`if D@desc(^name='Joe',descendant:P) and`
		`   P@person(^name:Name,^spouse:SP,person:C1) and`
		`   P1@person(^name=SP,person:C2) and`
		`   prolog{append(C1,C2,C), length(C,K)}`
		`then descendant(^married:'Yes',^kids:K,content:Name)`
		
		`if D@desc(^name='Joe',descendant:P) and`
		`   P@person(^name:Name,person:C) and`
		`   not P@person(^spouse:SP) and`
		`   prolog{length(C,K)}`
		`then descendant(^married:'No',^kids:K,content:Name)`

Queries		X-DEVICE
		The last two rules could also be written as:

```
if D@desc(^name='Joe',descendant:P) and
   P@person(^name:Name,^spouse:SP)
then descendant(^married:'Yes',content:Name)

if D@desc(^name='Joe',descendant:P) and
   P@person(^name:Name) and
   not P@person(^spouse:SP)
then descendant(^married:'No',content:Name)

if D@descendant(content:Name) and
   P@person(^name:Name,person ∍ C)
then D@descendant(^kids:count(C))

if D@descendant(^married='Yes',content:Name) and
   P@person(^name:Name,^spouse:SP) and
   P1@person(^name=SP,person ∍ C)
then D@descendant(^kids:count(C))
```

Sorting

```
if D@descendant(kids:K,content:Name)
then xml_result(result(descendant:ord_list(D-[K,Name]-[>,<])))
```