

## ASSIGNMENT - 1

Roll No. = CS22M055

NAME = MAHENDRA LACHETA.

### XPath:

1. Count the number of shops no floor\_no is 1.

`count(//floor/shop[floor_no="1"])`

Output:

☐ Pattern matching ☒ XPath 3.1 ☐ XQuery 3.1 ☐ CSS 3.0

```
1 count(//floor/shop[floor_no="1"])|
2
3
4
5
6
7
```

☐ disable auto refresh ☐ disable syntax highlight

Output Options: Node format::  Output format::

Compatibility:

Old languages: ☐ XPath 2.0 ☐ XPath 3.0 ☐ XQuery 1.0 ☐ XQuery 2.0

---

Result of the above expression applied to the above HTML file:

```
1 1
2
```

2. Find the floor\_no of shop name Calvin Kelin Jeans.  
`//shop/floor_no[./shop_name = "Calvin Kelin Jeans"]`

Output:

☐ Pattern matching ☒ XPath 3.1 ☐ XQuery 3.1 ☐ CSS 3.0 selectors ☐

```
1 //shop/floor_no[../shop_name = "Calvin Kelin Jeans"]
2
3
4
5
6
7
```

☐ disable auto refresh ☐ disable syntax highlighting

**Output Options:** Node format::  Output format::  ☐ Show types


**Compatibility:**

**Old languages:** ☐ XPath 2.0 ☐ XPath 3.0 ☐ XQuery 1.0 ☐ XQuery 3.0

---

Result of the above expression applied to the above HTML file:

```
1 0
2
```

A screenshot of a web browser window showing the XPath query interface. The browser's address bar is empty. The page content shows the query result as an empty set, represented by the symbol '0'.

3. Find the shop name whose rating is greater than 4.  
//shop/shop\_name[../@rating > "4.0"]

Output:

☐ Pattern matching ☒ XPath 3.1 ☐ XQuery 3.1 ☐ CSS 3.0 selector

```
1 //shop/shop_name[../@rating > "4.0"]
2
3
4
5
6
7
```

Submit Query ☐ disable auto refresh ☐ disable syntax highlighting

Output Options: Node format:: text Output format:: adhoc ☐ Show

Compatibility: Enable all extensions

Old languages: ☐ XPath 2.0 ☐ XPath 3.0 ☐ XQuery 1.0 ☐ XQuery

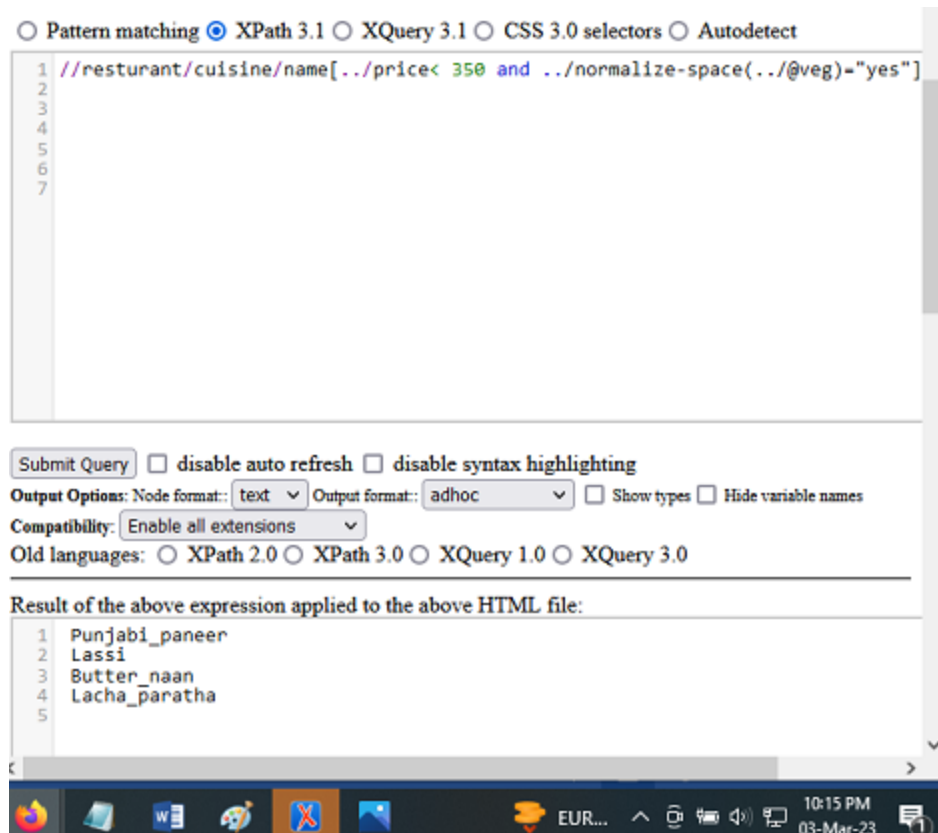
Result of the above expression applied to the above HTML file:

```
1 Calvin Kelin Jeans
2
```

4. Name of dishes whose price is less than 350 and whose restaurant type is vegetarian.

`//resturant/cuisine/name[../price< 350 and ../normalize-space(../@veg)="yes"]`

Output:



5. Find the sum of the price of the product Xiaomi 11T pro 5g and Mi Wired Headset.

`sum(//product/product_price[normalize-space(..product_name)="Xiaomi 11T pro 5g" or normalize-space(..product_name)="Mi Wired Headset"])`

Output:

Pattern matching

☒

XPath 3.1

XQuery 3.1

CSS 3.0 selectors

Autodetect

1

sum(//product/product\_price[normalize-space(..product\_name)="Xiaomi 11T pro 5g"

2

or normalize-space(..product\_name)="Mi Wired Headset"])

3

4

5

6

7

Submit Query

☐ disable auto refresh

☐ disable syntax highlighting

Output Options:

Node format::

text

Output format::

adhoc

☐ Show types

☐ Hide variable names

Compatibility:

Enable all extensions

Old languages:

☐ XPath 2.0

☐ XPath 3.0

☐ XQuery 1.0

☐ XQuery 3.0

Result of the above expression applied to the above HTML file:

1

37423

2

6. Find all the game names with an age limit of 10.
- ```
for $i in //games
return $i/game_name[normalize-space(..age_limit) > "10"]
```

Output:

☐ Pattern matching ☒ XPath 3.1 ☐ XQuery 3.1 ☐ CSS 3.0 selectors ☐ Autodetect

```
1 for $i in //games
2 return $i/game_name[normalize-space(..age_limit) > "10"]
3
4
5
6
7
8
```

☐ disable auto refresh ☐ disable syntax highlighting

Output Options: Node format:  Output format:  ☐ Show types ☐ Hide variable 1

Compatibility:

Old languages: ☐ XPath 2.0 ☐ XPath 3.0 ☐ XQuery 1.0 ☐ XQuery 3.0

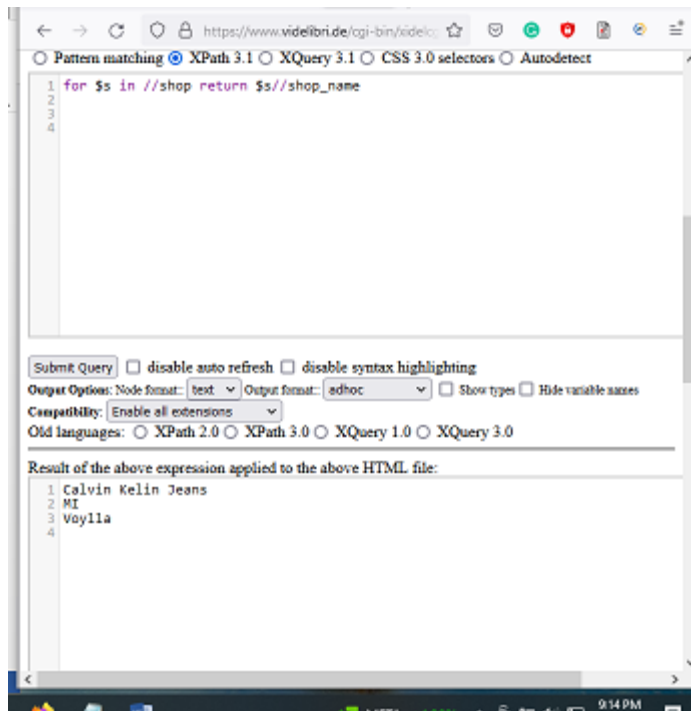
Result of the above expression applied to the above HTML file:

```
1 net_cricket
2 Snooker
3
```

## XQuery

1. return all the shop name  
for \$s in //shop return \$s/shop\_name

Output:



2. Return the product name whose price is greater than 10000.  
for \$x in //shop/product  
where \$x/product\_price>1000  
return \$x/product\_name

Output:

```
1 for $x in //shop/product
2 where $x/product_price>1000
3 return $x/product_name
4
5
6
7
```

Submit Query ☐ disable auto refresh ☐ disable syntax highlighting

Output Options: Node format:  Output format:  ☐ Show types ☐

Compatibility:

Old languages: ☐ XPath 2.0 ☐ XPath 3.0 ☐ XQuery 1.0 ☐ XQuery 3.0

---

Result of the above expression applied to the above HTML file:

```
1 T-Shirts
2 Jeans
3 Hoodie
4 Xiaomi 11T pro 5g
5 Sterling silver stylish Ring
6 925 Sterling silver CZ Flower Shape Diamond Pendant
7 Blue CZ Earring
8
```

3. Get the names and prices of all products with a rating of at least 4.0:
- ```
let $doc := //shop
return $doc//product[@rating >= 4.0]/(product_name, product_price)
```

Output:



```
1 let $doc := //shop
2 return $doc//product[@rating >= 4.0]/(product_name, product_price)
3
4
5
6
7
8
```

Submit Query ☐ disable auto refresh ☐ disable syntax highlighting

Output Options: Node format:: text Output format:: adhoc ☐ Show types ☐ Hide variable names

Compatibility: Enable all extensions

Old languages: ☐ XPath 2.0 ☐ XPath 3.0 ☐ XQuery 1.0 ☐ XQuery 3.0

Result of the above expression applied to the above HTML file:

```
1 T-Shirts
2 1500
3 Jeans
4 2000
5 Xiaomi 11T pro 5g
6 36999
7 Mi Wired Headset
8 424
9 Sterling silver stylish Ring
10 1569
11 925 Sterling silver CZ Flower Shape Diamond Pendant
12 3119
13 Blue CZ Earring
14 4119
```

4. Name of all the restaurants here at keyword can be used to count the iteration.

for \$x at \$i in //resturant

return

<resturant\_detail>

<resturant\_name>{\$i}. {data(\$x/res\_name)}</resturant\_name>

<resturant\_type>{data(\$x/@veg="yes")}</resturant\_type>

<dish>{data(\$x/cuisine/name)}</dish>

</resturant\_detail>

Output:

```

1 for $x at $i in //restaurant
2 return
3 <restaurant_detail>
4   <restaurant_name>{$i}. {data($x/res_name)}</restaurant_name>
5   <restaurant_type>{data($x/@veg="yes")}</restaurant_type>
6   <dish>{data($x/cuisine/name)}</dish>
7 </restaurant_detail>
8
9
10
11
12
13

```

Submit Query ☐ disable auto refresh ☐ disable syntax highlighting

Output Options: Node format: text Output format: adhoc ☐ Show types ☐ Hide variable names

Compatibility: Enable all extensions

Old languages: ☐ XPath 2.0 ☐ XPath 3.0 ☐ XQuery 1.0 ☐ XQuery 3.0

---

Result of the above expression applied to the above HTML file:

```

1
2 1. Wow Momo
3 false
4   veg steamed momos Burger chickenwings cold drink
5
6
7 2. Dominos
8 false
9   margerita pizza chicken pizza cheese garlic bread
10
11
12 3. punjabi_zaika
13 true
14   Punjabi_paneer paneer_achari Lassi Butter_naam Lacha_paratha
15

```

## 5. Name of restaurants and type it is veg or non-veg

```

for $x in //restaurant
return if ($x/@veg="yes")
then <veg>{data($x/res_name),data($x/@veg)}</veg>
else <non_veg>{data($x/res_name),data($x/@veg)}</non_veg>

```

Output:

☐ Pattern matching ☐ XPath 3.1 ☒ XQuery 3.1 ☐ CSS 3.0 selectors ☐ Autode

```
1 for $x in //restaurant
2 return if ($x/@veg="yes")
3 then <veg>{data($x/res_name),data($x/@veg)}</veg>
4 else <non_veg>{data($x/res_name),data($x/@veg)}</non_veg>
5
6
7
8
9
10
11 |
```

☐ disable auto refresh ☐ disable syntax highlighting

**Output Options:** Node format:  Output format:  ☐ Show types ☐ Hide

**Compatibility:**

**Old languages:** ☐ XPath 2.0 ☐ XPath 3.0 ☐ XQuery 1.0 ☐ XQuery 3.0

**Result of the above expression applied to the above HTML file:**

```
1 Wow Momo no
2 Dominos no
3 punjabi_zaika yes
4
```