Question E – Daniyar Nazarbayev [H00204990].

1.

fun function (x:bool) (y:bool) = true;

---

2. **16 unique truth tables.**

AND:

fun function (x:bool) (y:bool) = x andalso y;

OR:

fun function (x:bool) (y:bool) = x orelse y;

Tautology:

fun function (x:bool) (y:bool) = true;

Contradiction:

fun function (x:bool) (y:bool) = false;

x – statement 1:

fun function (x:bool) (y:bool) = x;

NOT x - negation of statement 1:

fun function (x:bool) (y:bool) = not(x);

NOT y – negation of statement 2:

fun function (x:bool) (y:bool) = not(y);

y – statement 2:

fun function (x:bool) (y:bool) = y;

Converse non-implication:

fun function (x:bool) (y:bool) = not(x) andalso y;

Converse implication:

fun function (x:bool) (y:bool) = x andalso not(y);

Implication (Conditional):

fun function (x:bool) (y:bool) = not(x) orelse y;

Non-implication:

fun function (x:bool) (y:bool) = x orelse not(y);

Biconditional (NXOR):

```
fun function (x:bool) (y:bool) = (not(x) orelse y) andalso (not(y) orelse x);
```

XOR:

```
fun function (x:bool) (y:bool) = (x andalso not(y)) orelse (y andalso not(x));
```

NAND:

```
fun function (x:bool) (y:bool) = not(x) andalso not(y);
```

NOR:

```
fun function (x:bool) (y:bool) = not(x) orelse not(y);
```

3.

They are both nameless functions, meaning that they can be called with “it” keyword.

They are of different types: int and real. Int has rounded numbers, Real has decimal point numbers, even rounded ones (example: 1.0).

4.

- First.

```
val a= [3,3,3,3,3];

val b= [4,3,3,3,3];

fun function1 (z) (x:int) = if z=[] andalso x=0 then false

else if z=[] then true

else if hd(z)=3 then true andalso function1 (tl(z)) (x+1) else false;

fun function2 (x) (y):bool = if x(y) (0)=true then true else false;
```

- Second.

```
val a= [3,3,3,3,3];

val b= [4,3,3,3,3];

fun function1 (z) = if z=[] then false

else if hd(z)=4 then true

else false orelse function1 (tl(z)) (y) (x+1);

fun function2 (x) (y):bool = if x(y)=true then true else false;
```

5.

fun function3 x = ();

maybe this one - fun function3 x = {};

Otherwise I think it’s the only one, since the only possible way to get unit is by calling an empty tuple.

6.

fun function4 x = (x, x);

I think it’s the only one since there is only one parameter and end result is a tuple with two identical parameters.

7.

fun function5 (x) (y) = (x,y);

8.

2 functions – in SML there are only single argument functions.

it could be written as fn x => fn y => (x,y);

9.

fun func (a) (b) = (a:'a, b:'a);

10.

2 functions - in SML there are only single argument functions.

it could be written as fn a => fn b => (a:’a,b:’a);

11.

fun func ((f: 'a->'b), x:'a) = f(x);

12.

1 function since both arguments are in a tuple, and a tuple is considered as single argument to the fanction.

13.

fun func (f1:'a->'b) (f2:'b->'c) = f2 o f1;

14.

3 functions – 2 fanctions as parameters, but it also returns a fanction.

fn f1:’a->’b => f2:’b->’c => f2 o f1;

15.

fn x:'a => fn f2:'b->'c => fn f3:'a->'b => f2 o f3;

16.

exception exc;

fun function () = raise exc;
