Functional Programming Assignment 1

Benjamin Bissett s184449 - Kristofer Kandle s184352 - Mikkel Christophersen s184393

Task 1

1.

We can see that collect gives two polymorphic types in f#:
('a -> 'b list) -> 'a list -> 'b list
There is no specific inferrence of type in the function and as stated in the week two slide 16, it proves this is the principle type of the function as any declaration of types of variable would bind a and b to them and thus would be an instance of the polymorphic type.
Without inferring a:float/int the function collect can assume the type in lists given aslong as the type is constant throughout the list.

```
An example: collect (fun(a,b)->[a..b]) [(1.0,3.0);(4.0,7.0);(8.0,8.0)];;
            val it: float list = [1.0;2.0;3.0;4.0;5.0;6.0;7.0;8.0]

            collect (fun(a,b)->[a..b]) [(1,3);(4,7);(8,8)];;
            val it: int list = [1;2;3;4;5;6;7;8]
```

2.

```
collect(fun(a,b)->[a..b]) [(1,3) ; (4,7) ; (8,8)]
⇒fun(1,3) @ collect (fun(a,b)->[a..b])[(4,7) ; (8,8)]
⇒[1;2;3] @ collect (fun(a,b)->[a..b])[(4,7) ; (8,8)]
⇒[1;2;3] @ fun(4,7) @ collect (fun(a,b)->[a..b])[(8,8)]
⇒[1;2;3] @ [4;5;6;7] @ collect (fun(a,b)->[a..b])[(8,8)]
⇒[1;2;3;4;5;6;7] @ collect (fun(a,b)->[a..b])[(8,8)]
⇒[1;2;3;4;5;6;7] @ fun(8,8) @ collect (fun(a,b)->[a..b])[ ]
⇒[1;2;3;4;5;6;7] @ [8] @ collect (fun(a,b)->[a..b])[ ]
⇒[1;2;3;4;5;6;7;8] @ collect (fun(a,b)->[a..b])[ ]
⇒[1;2;3;4;5;6;7;8] @ []
⇒[1;2;3;4;5;6;7;8]
```

3.

For collect in the expression - collect (fun(a,b)->[a..b]) [(1,3);(4,7);(8,8)];; - the type is val it : int list = [1; 2; 3; 4; 5; 6; 7; 8] therefore of the int type. This is an instance of the most general type shown in part 1 because f# has recognized that the values within the brackets are integers, and has assigned the list as a list of integers, thereby removing the polymorphic type declaration from the variables.