Exercise 1 (5 pts) Exercise 5.2.1 on page 59 in the textbook.

Exercise 2 (5 pts) Exercise 5.2.4 on page 61 in the textbook.

Exercise 3 (5 pts) Exercise 5.2.5 on page 62 in the textbook.

Exercise 4 (5 pts) Exercise 5.2.7 on page 63 in the textbook.

Exercise 5 (30 pts) Let us use the following datatypes term and term1 for representing untyped (pure) \( \lambda \)-terms and their \( \alpha \)-normal forms, respectively:

```ocaml
datatype term =
  TmVar of string | TmLam of (string, term) | TmApp of (term, term)

datatype term1 =
  TmVar1 of string | TmInd1 of int | TmLam1 of term1 | TmApp1 of (term1, term1)
```

- (10 pts) Please implement a function nf_alpha that translates a given \( \lambda \)-term into its alpha-normal form:

```ocaml```
```java
extern fun nf_alpha (t: term): term1
```

- (20 pts) Please implement a function subst of the following type:

```ocaml```
```java
extern fun subst (t: term, x: string, s: term): term
```

Given a term \( t \), a variable \( x \) and a term \( s \), \( \text{subst}(t, x, s) \) should return \( t[x \mapsto s] \).

Exercise 6 (20 pts) The following declared dataprop F91 encodes the MacCarthy’s 91-function:

```ocaml```
```java
dataprop F91 (int, int) =
  | F91def1 (91, 91)
  | {i:int | i <= 100; i <> 91} {r1,r2:int} F91def2 (i, r2) of (F91 (i+11, r1), F91 (r1, r2))
  | {i:int | 101 <= i} {r:int} F91def3 (i, r) of F91 (i-10, r)
```

1
Given integers $i$ and $r$, if a proof value can be assigned the type $F91(i, r)$, then $f91(i) = r$, where $f91$ is formally defined in Assignment 1. Please construct a proof function $f91\_lemma$ in ATS that is declared as follows:

```
prfun f91_lemma {i,r:int} (pf: F91 (i, r)): [r==91] void
```

**Exercise 7 (40 pts)** The definition of Braun trees is encoded into the following declared dataprop $isBraun$:

```
datasort bt = B of (bt, bt) | E of ()
dataprop isBraun (bt) =
| {t1,t2:bt} {s1,s2:nat | s2 <= s1; s1 <= 1 + s2} isBraun_B (B (t1, t2)) of (isBraun t1, isBraun t2, btsz (t1, s1), btsz (t2, s2))
| isBraun_E (E ()) of ()
```

Please construct a proof function $brauntree\_height\_lemma$ in ATS that is declared as follows:

```
prfun brauntree_height_lemma {t1,t2:bt} {h,h1:nat} (pf0: isBraun (B (t1, t2)), pf1: btht (B (t1, t2), h), pf2: btht (t1, h1)) : [h == h1+1] void
```

Please also construct a proof function $brauntree\_size\_height\_lemma$ in ATS that is declared as follows:

```
prfun brauntree_size_height_lemma {t:bt} {s,h,n:nat} (pf0: isBraun (t), pf1: btsz (t, s), pf2: btht (t, h), pf3: POW2 (h, n)) : [n <= s + s + 1] void
```

Note that the dataprops $btsz$, $btht$ and $POW2$ are all declared in a file available on-line.