1)

Topic	Scheme	Haskell
Datatypes	It supports the basic numbers(int and float) and values(characters,string), functions and boolean. Also supports user defined types.	It supports the basic values like integers,boolean, characters,string and booleans. Also supports user defined types.
Method of Operation	Works with prefix types	Doesn’t need to be in prefix style. It can work in both styles.
Operators	Supports arithmetic(+,-,*,/,rem,mod,incf,decf),logical(and,or,not), relational(<,>,==,!=,<=,>=,<>) operators	Supports arithmetic(+,-,*,/,++,--,!!,..,**,^,^^),logical(&&,||,!), relational(<,>,==,!=,<=,>=,<>) operators. In addition it also supports ->,<-,\\which are referncing operators.It also supports =>,(),>>,>>= and so many small operators
Control Flow	It uses if and case mainly in case of conditional checking.	Supports if and case type of implementation for control flow.
File handling and I/O operations	Both these are handled efficiently by the software.	Both these are handled efficiently by the software.
Functions and Looping	Functions are bounded with (). Doesn’t support curried functions. Uses tail recursions and normal recursions.	Functions are not necessarily bouned within (). Supports curried functions. Uses lazy eavaluation to compute recursions.

Typing	Dynamic typing Duck typing	Strong static typing (though with type inference) Parametric polymorphism
Syntax	S-exprs,Macros,Uniform syntax,semi-concise	Infix notation encouraged,Template -Haskell (A way of doing meta-programming in haskell),Consistent syntax,Very terse
Concepts	Pragmatically functional,Code as data - Data as code,Avoid complexity,Syntactical and procedural abstraction	Purely (and still pragmatically) functional,Types and Static checking,Functional and Data abstraction, Encourage multiple layers of abstraction

---

2)

Type Checking:
    In case of semantic type checking here is an advantage of having case sensitivity as 2 different datatypes might want to have a same name and hence case insensitivity would undermne such a definition due to type inconsistancy.
    Thus case senstivity is a boon when type checking is concerned.

```
    Eg: byte rgbData[MAX];
        typedef struct _bdata
        {
            int   nID;
            DWORD dwFlags;
        } BDATA;

        BDATA rgbdata[MAX];
```

Here rgbData and rgbdata must be differentiated.

Compiler Complexiy:
In case insensitive scenario all the variables must be convered to a common case.
Thus is an additional time consuming process in case of case insensitivity.
Hence Case insentitivity increases compiler complexity.
Eg: Hello and HEllO must be converted to common case say HELLO and the compiled.

Reliability:
When we have case insensitivoty, reliability decreases as the error pointed out by the compiler becomes confusing as all the variables of same sructure are reported by a single cased letter by the compier due to the above conversion process.
Eg: If a program contains expected and Expected, it would throw an error line error in EXPECTED which is confusing as to which expected the compiler is referring to in the current context.

---

3)

C

String Libraries
It has limited functionalities for strings. These include functions for computing length,comparing strings,copying strings and concatinating strings.

Arrays
- It uses 0 based indexing.
- It doesn’t have any special symbols for declarations and usage.
- Arrays cannot be autofilled.
- Doesn’t have special functions to select range of values and similar sort of functionality.
- Doesn’t support hashing.

ADA

String Libraries
It provides more functionalities with respect to strings.It provides those as provided by C and also provides more functionalities like maps and hash values to strings and also provides slicing and selection methods which are much simpler and easier to access compared to C.

Arrays

- It used 1 based indexing.
- It doesn’t have any special symbols for declarations and usage.
- Arrays can be autofilled.
  ```
      array (1 .. 10) of integer;
  ```
- Has special functions to select range of values and similar sort of functionality.
- Supports hashing.

PERL

String Libraries

It provides more functionalities with respect to strings.It provides those as provided by C and also provides more functionalities like hash values to strings and also provides slicing and selection methods which are much simpler and easier to access compared to C.

Arrays

- It uses 0 based indexing.
- It uses “@” for referring to an array and “$” for referring to the value of an array.
- Arrays can be autofilled.
  ```
    @nums = (1..20);
  ```
- Has special functions to select range of values and similar sort of functionality
- Supports hashing and is referenced using “#”.

---

4)

```
STRING
test.adb:42:31: expected type "Standard.Integer"
test.adb:42:31: found a string type
INTEGER
test.adb:42:31: value not in range of type "Standard.Integer"
FLOAT
Average is 4.29497E+09
```
