Appendix 2 - Function List

A full list of functions described in the "Functions" chapters

A * to the left indicates that declaring the types of the arguments may lead to inline code - see section on coding efficiency.

main list functions

```
* list[vals ] or vals  -> list ;
* splice[list ] or .list  -> vals ;
```

other list functions

```
  join[lists ]-> list ;
  nth[int ,list ]-> val ;
  nthcdr[int ,list ]-> list ;
  length[list ]-> int ;
  ismember[val ,list ]-> bool ;
  nil[] ->;
  prepend[val ,list ]-> list ;
  append[val ,list ]-> list ;
  car[list ]-> val ;
  cdr[list ]-> list ;
```

equality

```
* eq[val ,val ] or val = val -> bool ;
* ne[val ,val ] or val != val -> bool ;
```

output

```
  outs[ints ] -> ; (be careful with this)
  outnum[vals ] -> ;
  outbase[int ,vals ] -> ;
  outstring[vals ] -> ;
  prettyprint[vals ] -> ;
  prettyprint16[vals ] -> ;
  outand[vals ] -> vals

  SYSresidue[vals ] -> ; (can be user defined)
```

basic arithmetic

```
* add[ints ]  or  int + int  -> int ;
* mul[ints ]  or  int * int  -> int ;
* neg[int ]  or  -int  -> int ;
* sub[int ,int ]  or  int - int  -> int ;
* div[int ,int ]  or  int / int  -> int ;
* mod[int ,int ]  or  int % int  -> int ;
```

comparison

```
* ge[int ,int ]  or  int >= int  -> bool ;
* gt[int ,int ]  or  int > int  -> bool ;
* le[int ,int ]  or  int <= int  -> bool ;
* lt[int ,int ]  or  int < int  -> bool ;
  ge[char ,char ]  or  char >= char  -> bool ;
  gt[char ,char ]  or  char > char  -> bool ;
  le[char ,char ]  or  char <= char  -> bool ;
  lt[char ,char ]  or  char < char  -> bool ;
```

max/min

```
  abs[int ] -> int ;
* max[ints ] -> int ;
* min[ints ] -> int ;
  maxint[] -> $7FFFFFFF;
  minint[] -> -$7FFFFFFF;
```

random numbers

```
  random[] -> int ;
```

boolean

```
* and[bools ]  or  bool & bool  -> bool ;
* or[bools ]  or  bool | bool  -> bool ;
* not[bool ]  or  !bool  -> bool ;
```

bit operations

```
* bitand[ints ] -> int ;
* bitor[ints ] -> int ;
* bitnot[int ] -> int ;
  hi[int ] -> int ;
  lo[int ] -> int ;
* shift[int ,int ] -> int ;
  bitff1[int ] -> int ;
  bitff0[int ] -> int ;
  bitcount[int ] -> int ;
```

trees

```
  tss[ints ,tree ,val <,fill >]-> ints ,tree ,val ;
  treeslicesplice = tss
  treetake[ints ,tree ,val <,fill >]-> val ;
  treeput[ints ,tree ,val <,fill >]-> tree ;
```

sets

```
  osetaddmem[int ,set ] -> set ;
  osetremmem[int ,set ] -> set ;
  osetismem[int ,set ] -> bool ;
  osetcount[set ] -> int ;
  osetfindfirst[set ] -> int ;
  osetunion[set ,set ] -> set ;
  osetintersect[set ,set ] -> set ;
  osetdiff[set ,set ] -> set ;
  osetfill[int ] -> set ;
```

glue

```
  first[vals ] -> val ;
  second[vals ] -> val ;
  third[vals ] -> val ;
```

misc numerical

```
  tickcount[] -> int ;
  Uniqnum[] -> int ;
```

input

```
  button[] -> bool ;
  buttonpress[] -> bool ;
```

****

The function from here on down are old versions that only work with integers and lists.

****

input

```
  getkey[] -> int ;
  getline[] -> ints ;
```

parsing characters

```
  iswhitespace[int ] -> int ;
  isalpha[int ] -> int ;
  isdigit[int ] -> int ;
```

outstring

```
  outstring[vals ] -> int ;
```