TEST 1

```
kotoba program1;

declare number x, number arr[4.0], word w, bool b, sentence s, sentence ss;

function number myfunc(number y){
   declare number x;
   if(y > 2.0){
     set  y = y + 1.0;
   }else{
      set y = y * 2.0;
   }
   return y;
}

begin
{
   kread(w);
   if(!(x < 1.0)){
      kprint(s,1.0);
   }
   if((x < 1.0) & (s == ss)){
      s.wordCount();
   }

   while(x > 10.0){
     set  x = x - 1.0;
   }

   do{
     set x = x / 2.0;
   }while(x > 20.0);
}
end
```

TEST 2

```
kotoba program1;
    declare number x, number z, bool b, sentence s;

    function number myfunc(number y, bool w){
    declare number x;
    if(y > 2.0) {
       set  y = x + 1.0;
    }
    return y;
    }

    begin
    {
```

```
        set b = false;
        call myfunc(x, b);
        set x = x * 2.0;
    }
    end
```

TEST 3

```
kotoba program1;
declare number x, bool b, word w, sentence s;
begin
{
    set x = 15.0;
    set b = false;
    set w = "Hello";
    set s = "Welcome to Kotoba.";
    kprint(x);
    kprint(b);
    kprint(w);
    kprint(s);
}
end
```

TEST 4

```
kotoba program1;

           declare number x, number f;

           function number sum(number x){
               return (x + 1.0);
           }

           function number mult(number x){
               return (x * 2.0);
           }

           function number uno(number n) {
               declare number a, number b;

               set a = call sum(n);
               kprint(a);
```

```
        set b = call mult(n);
        kprint(b);

        return a;
    }

    begin
    {
        set x = 3.0;
        set f = call uno(x);
        kprint(f);
    }
    end
```

Fibonacci Recursive

```
kotoba program1;

    declare number x, number f;

    function number fib(number n) {
        declare number a, number aAux, number b, number bAux;
        if(n < 2.0){
            return n;
        }else{
            set aAux = n - 1.0;
            set bAux = n - 2.0;
            set a = call fib(aAux);
            set b = call fib(bAux);
            return (a + b);
        }
    }

    begin
    {
        set x = 3.0;
        set f = call fib(x);
        kprint(f);
    }
    end
```

Fibonacci Cycle

```
kotoba program1;

          declare number x;

          function void fib(number n) {
               declare number a, number b, number aux, number i;

               set a = 0.0;
               set b = 1.0;
               set i = 2.0;
               kprint(a);
               kprint(b);

               while(i < (n + 1.0)) {
                   set aux = a + b;
                   set a = b;
                   set b = aux;
                   set i = i + 1.0;
                   kprint(aux);
               }


               return "void";
      }


          begin
          {
              set x = 7.0;
              call fib(x);
          }
          end
```

Recursion TEST

```
kotoba program1;

          declare number x, number f, number y;

          function number test(number n) {
```

```
    declare number a, number aAux, number b, number bAux;
    if(n < 2.0){
        return n;
    }else{
        set aAux = n - 1.0;
        set a = call test(aAux);
        return (a + 1.0);
    }
}

begin
{
    set x = 3.0;
    set f = call test(x);
    kprint(f);
}
end
```

Factorial Recursive

```
kotoba program1;
    declare number f;

     function number factorial(number n) {
         declare number nAux, number param;

         if (n == 1.0){
             return 1.0;
         }else{
             set param = n - 1.0;
             set nAux = call factorial(param);
             return (n * nAux);
         }
     }

begin
{
    set f = call factorial(5.0);
    kprint(f);
}
end
```

Factorial Cycle

```
kotoba program1;
      declare number factorial, number n, number i;

      begin
      {
           set factorial = 1.0;
           set n = 8.0;
           set i = 2.0;

           while(i < n + 1.0){
               set factorial = factorial * i;
               set i = i + 1.0;
           }

           kprint(factorial);

      }
      end
```

Arrays

```
kotoba program1;
       declare word w[4.0], number n, sentence s, number arr[5.0];

       begin
       {
           set s = "The dog is running.";
           set arr = {2.0, 34.5, 67.56, 4.1, 0.12};

           set w = call s.tokenize();

           set n = 0.0;

           set s = w[n] + w[1.0];

           kprint(w[n]);
           kprint(arr[3.0]);
           kprint(s);
       }
```

```
    end

kotoba program1;
    declare word w[4.0], number n, sentence s, number arr[5.0], number len;

    begin
    {
        set s = "The dog is running.";
        set arr = {2.0, 34.5, 67.56, 4.1, 0.12};

        set w = call s.tokenize();

        set n = 0.0;

        set len = call w.size();

        while(n < len) {
            kprint(w[n]);
            set n = n + 1.0;
        }

        set s=w[0.0]+w[1.0];

        kprint(s);

        set n = 0.0;
    }
    end
```

Sort and Exists

```
kotoba program1;
    declare word w[4.0], number arr[5.0], bool flag;

    begin
    {
        set w = {"orange", "apple", "grape", "banana"};
        set arr = {2.0, 34.5, 67.56, 4.1, 0.12};

        set flag = call w.exists("apple");
        if(flag){
```

```
            call w.sortWords();
            call arr.sortNumbers();
            kprint(w[1.0]);
            kprint(arr[3.0]);
        }else{
            kprint(w[1.0]);
            kprint(arr[3.0]);
        }

}
end
```