CREST Examples

- Basic Examples
- Function Examples
- Limitation Examples

Basic Example 1

```
// Hello CREST example.
// This example shows how to define a symbolic variable.
#include <crest.h> // for CREST
#include <stdio.h>

int main(){
    int x;
    CREST_int(x); // Define x as a symbolic input.

    printf("x = %d₩n", x);
    if (x > 100){
        printf("x is greater than 100₩n");
    }else{
        printf("x is less than or equal to 100₩n");
    }
    return 0;
}
```

Basic Example 2

```
// Another Hello CREST example.
// CREST can handle linear integer arithmetic expression
// and nested condition statements
#include <crest.h>
#include <stdio.h>
int main()
    char x, y;
    CREST_char(x);
    CREST_char(y);

    printf("x, y = %d, %d₩n", x, y);
    if (2 * x == y){
        if (x != y + 10) printf("Fine here₩n");
        else             printf("ERROR₩n");
    }
    return 0;
}
```

Basic Example 3

```
// Symbolic value propagation example.
// In an assign statement, if RHS has a symbolic variable and the symbolic
// variable is used in linear integer arithmetic expression, LHS will be
// a symbolic variable
#include <crest.h>
#include <stdio.h>

int main(){
    int x, y;
    CREST_int(x);

    printf("x = %d₩n", x);
    y = 2 * x + 3;

    if (y == 7)         printf("y(=2x+3) is 7₩n");
    else        printf("y(=2x+3) is NOT 7₩n");
}
```

Basic Example 4

```
// Long symbolic path formula generated due to a loop
#include <crest.h>
#include <stdio.h>

int main(){
    int i, x;
    CREST_int(x);

    for (i=0; i < x; i++) {
        printf("i=%d₩n",i);
        if ( i == 3) {
            printf("i becomes 3 finally₩n");
            break;
        }
    }
}
// use print_execution to print a symbolic execution path formula
```

Function Example 1

```
// Simple function example
// Symbolic variable can be passed into a function.
#include <crest.h>
#include <stdio.h>

void test_me(char x, char y){
    // body of test_me is same to basic2 example
    if (2 * x == y){
        if (x != y + 10){
            printf("Fine here₩n");
        }else{
            printf("ERROR₩n");
        }
    }
}

int main(){
    char a, b;

    CREST_char(a);
    CREST_char(b);

    printf("a, b = %d, %d₩n", a, b);
    test_me(a, b);
    return 0;
}
```

Function Example 2

```
// Another simple function example.
// A function can return a symbolic value
#include <crest.h>
#include <stdio.h>

int sign(int x){     return (x >= 0);}

int main(){
     int a;
     CREST_int(a);
     printf("a = %d₩n", a);

     if (sign(a) == 0)        printf("%d is negative₩n", a);
     else          printf("%d is non-negative₩n",a);
     return 0;
}
```

Function Example 3

```
// Recursive function example.
// CREST can handle a recursive function.
// A recursive function can generate infinite # of iterations.
#include <crest.h>
#include <stdio.h>

unsigned int fac(unsigned int n){
    if (n == 0) return 1;
    else return n * fac(n-1);
}

int main(){
    unsigned int a;
    CREST_unsigned_int(a);
    printf("a = %u₩n", a);

    if (fac(a) == 24)         printf("Reach!₩n");
    return 0;
}
```

Limitation 1: No External Binary Library

```
// External library example.
// When a target program calls an external library function,
// CREST may occur 'prediction failure' error since CREST
// does not know a body of the external function
#include <crest.h>
#include <stdio.h>
#include <stdlib.h>
int main(){
    int x;
    CREST_int(x);
    printf("x == %d₩n");
    if (x == abs(x)){// Generate symbolic path formula using
                     // a concrete return value (i.e., x == 0)
        printf("x >= 0₩n");
    }else{
        printf("x <= 0₩n");
    }
    return 0;
}
```

Limitation 2: No Non-linear Arithmetic Expression

```
// CREST cannot handle a non-linear arithmetic expression
#include <crest.h>
#include <stdio.h>
int main(){
    int x;
    CREST_int(x);
    printf("x = %d₩n", x);

    if ((x+1)*(x+1) == 4){// Generate symbolic path formula
                          // using a concrete value (i.e., x+1==4)
    // if ((x+1)*(x+1) == (long long) 4){ // Difference?
        printf("ERROR₩n");
    }else{
        printf("Fine₩n");
    }
    return 0;
}
```

Limitation 3: Real Numbers

```
// Real numbers cannot be
// fully handled by CREST
#include <crest.h>
#include <stdio.h>
int main(){
    int x;
    CREST_int(x);
    printf("x = %d₩n", x);

    if (x + 2.3 == 4 + 2.4){
        printf("x is 4₩n");
    }else{
        printf("x isn't 4₩n");
    }
    return 0;
}
```

```
$ run_crest float3 100 -dfs
Iteration 0 (0s): covered 0 branches [0 reach
funs, 0 reach branches].
x = 0
x isn't 4
Iteration 1 (0s): covered 1 branches [1 reach
funs, 2 reach branches].
x = 4
x isn't 4
Iteration 2 (0s): covered 1 branches [1 reach
funs, 2 reach branches].
Expected branch to take is 3 but 4 is executed
at the flipping point
Prediction failed!
elapsed time in Yices = 0.0001248
opt1: 0, opt2: 0
```

Limitation 4: No Symbolic Array

```
// Array cannot be declared symbolically.
// Instead, each element can be declared symbolically
#include <crest.h>
#include <stdio.h>
int main(){
    int i;
    int array[4];

    // CREST_int(array); // NOT WORKING
   for(i=0; i < 4; i++)
       CREST_int(array[i]);

    if (array[1] == 3)
          printf("array[1] is 3₩n");
    else printf("array[1] is not 3 but ₩%d₩n",array[1]);
}
```

Limitation 5: No Symbolic Dereference

```
// Symbolic dereference is not supported.
// If an array index is a symbolic variable, CREST does not generated
// a corresponding symbolic path formula
#include <crest.h>
#include <stdio.h>
int main(){
    int x;
    int array[4];

    CREST_int(x);
    printf("x = %d₩n", x);
    array[0] = 0;
    array[1] = 1;
    array[2] = x;
    array[3] = 4;

    if (array[x-1] == 3)  printf("ERROR₩n");
    else          printf("Fine₩n");
}
```

Should check the following Symbolic path formula

(x==1 && array[0] ==3) ||
(x==2 && array[1] ==3) ||
(x==3 && array[2] ==3) ||
(x==4 && array[3] ==3)

Partial Solution for Limitation 5

```
#include <crest.h>
#include <stdio.h>
#define ENUM_4(array, index, ret) ₩
do{ ₩
    switch(index){ ₩
        case 0: ₩
            ret = array[0]; ₩
            break;₩
        case 1: ₩
            ret = array[1]; ₩
            break; ₩
        case 2: ₩
            ret = array[2]; ₩
            break; ₩
        case 3: ₩
            ret = array[3]; ₩
            break; ₩
    } ₩
}while(0);
```

```
int main(){
    int x, tmp;
    int array[4];

    CREST_int(x);
    if (x < 1 || x > 4){ exit(0); }
    printf("x = %d₩n", x);
    array[0] = 0;
    array[1] = 1;
    array[2] = x;
    array[3] = 4;
    // tmp = array[x-1]
    ENUM_4(array, x-1, tmp);

    if (tmp/*array[x-1]*/ == 3){
        printf("ERROR₩n");
    }else{
        printf("Fine₩n");
    }
}
```