1. Cast(explicit-conversion):

Summary:Test three types of cast(int to short, long long to int, signed to unsigned), each with lossy values.

Source File: cast.c

```
#include "stdlib.h"

#include "stdint.h"

#include "stdio.h"

int main()

{

   printf("Test:cast:¥n");

   printf("Test int to short conversion with value loss¥n");

   int a=INT32_MAX;

   short c=(short)a;

   printf("int=%d, short=%d¥n",a,c);

   printf("End Test int to short conversion¥n");

    printf("Test long long to int conversion with value loss¥n");

   long long along=INT64_MAX;

   int cint=(int)along;

   printf("longlong=%d, int=%d¥n",a,c);

   printf("Test signed to unsigned cast¥n");

   printf("End Test longlong to int¥n");

   int sint=-1;

   unsigned int uint=(unsigned int)sint;

   printf("signed int=%d, unsigned int=%u¥n",sint,uint);
```

```
    printf("End Test signed int to unsigned int¥n");
}
```

```
Flags:(-fioc-explicit-conversion) clang -fioc-explicit-conversion cast.c -o cast
```

Result: No detection(?)

```
Test:cast:

Test int to short conversion with value loss

int=2147483647, short=-1

End Test int to short conversion

Test long long to int conversion with value loss

longlong=2147483647, int=-1

Test signed to unsigned cast

End Test longlong to int

signed int=-1, unsigned int=4294967295

End Test signed int to unsigned int
```

2. Cast(implicit-conversion):

Summary:Test three types of implicit cast(int to short, long long to int, signed to unsigned), each with lossy values.

Source File: implicit_cast.c

```
#include "stdlib.h"
```

```
#include ″stdint.h″

#include ″stdio.h″

int main()

{

        printf(″Implicit Test:cast:¥n″);

        printf(″Test int to short conversion with value loss¥n″);

        int a=INT32_MAX;

        short c=a;

        printf(″int=%d, short=%d¥n″,a,c);

        printf(″End Test int to short conversion¥n″);

        printf(″Test implicit long long to int conversion with value loss¥n″);

        long long along=INT64_MAX;

        int cint=along;

        printf(″longlong=%d, int=%d¥n″,a,c);

        printf(″End Test longlong to int¥n″);

        printf(″Test signed to unsigned cast¥n″);

        int sint=-1;

        unsigned int uint=sint;

        printf(″signed int=%d, unsigned int=%u¥n″,sint,uint);

        printf(″End Test signed int to unsigned int¥n″);



}
```

Flags:(-fioc-implicit-conversion) clang -fioc-implicit-conversion implicit_cast.c -o implicit_cast

Result: No detection(?)

```
Implicit Test:cast:

Test int to short conversion with value loss

int=2147483647, short=-1

End Test int to short conversion

Test implicit long long to int conversion with value loss

longlong=2147483647, int=-1

End Test longlong to int

Test signed to unsigned cast

signed int=-1, unsigned int=4294967295

End Test signed int to unsigned int
```

3.Integer Overflow:

Summary:Test INT32_MAX+1 in the code

Source File: overflow.c

```
#include ″stdlib.h″

#include ″stdint.h″

#include ″stdio.h″

int main()
```

```
{
        printf("Test Integer Overflow:¥n");
        int a=INT32_MAX;
        int b=a+1;
        printf("a=%d, a+1=%d¥n",a,b);
        printf("End Test integer overflow¥n");

}
```

Flags:-fioc-signed(clang -fioc-signed overflow.c -o overflow)

Result: Detected(Runtime)

```
overflow.c:8:16: runtime error: signed addition overflow [ expr = '+', lval = (sint32) 2147483647, rval = (sint32) 1 ]
a=2147483647, a+1=-2147483648
End Test integer overflow
```

4. Integer Underflow(signed):

Summary: Test INT32_MIN-1 in the code

Source File: underflow.c

```
#include "stdlib.h"
#include "stdint.h"
#include "stdio.h"
```

```
int main()

{

        printf("Test Integer Underflow¥n");

        int a1=INT32_MIN;

        int b1=a1-1;

        printf("a=%d, a-1=%d",a1,b1);

        printf("End Test Integer Underflow¥n");

}
```

Flags: -fioc-signed (clang -fioc-signed underflow.c -o underflow)

Result:

```
Test Integer Underflow

underflow.c:8:18: runtime error: signed subtraction overflow [ expr = '-', lval = (sint32) -2147483648, rval = (sint32) 1 ]

a=-2147483648, a-1=2147483647End Test Integer Underflow
```

5.Unsigned to Signed Cast(Implicit):

Summary:Test Cast from UINT32_MAX to int

Source File:unsignedtoint.c

```
#include "stdio.h"

#include "stdlib.h"

#include "limits.h"

int main()
```

```
{
 printf("Start conversion from unsigned to signed:¥n");
 unsigned u=UINT_MAX;
 int i=u;
 printf("the original unsigned integer is: %u, the converted signed integer is: %d¥n", u, i);
 printf("End test");
 return 0;
}
```

Flags:-fioc-implicit-conversion(clang -fioc-implicit-conversion unsignedtoint.c -o unsignedtoint)

Result:Detected

```
Start conversion from unsigned to signed:
unsignedtoint.c:8:13: runtime error: value lost in conversion of '2147483647' from 'int' (int) to 'unsigned int' (unsigned int)
the original unsigned integer is: 4294967295, the converted signed integer is: -1
```

6.Divide by Zero:(seems IOC doesn't handle that because clang has already do something with that undefined behavior)

Summary:Test Divided by Zero

SourceFile:dividebyzero.c

```
#include "stdio.h"
int main()
```

```
{
 printf("Input the number you want to divide:¥n");
 int i=0;
 scanf("%d",&i);
 printf("divide 100 by %d", i);
 float f=(float)100/i;
 printf("the result is %f¥n",f);
}
```

Flags:-fcatch-undefined-behavior(clang -fcatch-undefined-behavior dividebyzero.c -o dividebyzero)

Result:Detected

```
Input the number you want to divide:
0
非法指令 (核心已转储)(illegal instruction)
```

7.MallocError(a type of implicit cast from signed integer to unsigned integer)

Summary: the malloc function's parameter is -1, it will be converted to a very large unsigned integer

SourceFile:mallocerror.c

```
#include "stdlib.h"
#include "stdio.h"
int main()
{
    printf("parameter for (sizeof(char)*a) is a=-1¥n");

    int a=-1;

    char * cp;

    if(a<100)

    {

        cp=(char *)malloc(sizeof(char)*a);

    }

    return 0;

}
```

Flags:-fioc-implicit-conversion(clang -fioc-implicit-conversion mallocerror.c -o mallocerror)

Result:not detected

```
./mallocerror parameter for (sizeof(char)*a) is a=-1
```