Bilaga Felhantering

Interpretatorn ger stöd för 26 felmeddelanden. Felmeddelandena täcker inte alla fel som kan uppstå. I de fall ett fel identifieras och ett passande felmeddelande inte kan identifieras kommer felmeddelande CODE_80 att skrivas ut. Vidare garanterar inte interpretatorn att alltid skriva ut felmeddelanden vid en krasch, det som avgör är om kraschen existerar utanför eller i felhanteringssystemet. Nedan följer en lista på samtliga felmeddelanden:

CODE_10: Overflow: Buff overflow, cannot access a memory address that has not been allocated.

CODE_11: Overflow: Stackoverflow, cannot write to memory outside the Stack.

CODE_12: Initialization Stack: Stack cannot be bigger than Global.

CODE_13: Initialization Stack: You cannot create a Stack that is smaller than one byte.

CODE_14: Initialization Global: Global need to be minimum one byte

CODE_15: Global: You can only initilize Global once.

CODE_16: Stack: You can only create one Stack.

CODE_30: Alias: You cannot create alias on Stack.

CODE_31: Alias: Name need to contain least two character and start with a letter, symbols are not allowed.

CODE_32: Alias: Cannot find name on Global.

CODE_34: Alias: Missing ':'.

CODE_35: Alias: address need to start with '#' and fallowing character need to be digits.

CODE_3510: Alias: Datatype int, Int cannot contain letters.

CODE_3511: Alias: Datatype int, Int can’t be bigger than INT_MAX.

CODE_40: Struct: Struct name missing.

CODE_41: Struct: Missing open bracket.

CODE_42: Struct: Missing close bracket.

CODE_43: Struct: Missing open and close brackets.

CODE_44: Struct: struct name length is maximum 50 characters, or there is no open and close brackets.

CODE_50: Subroutine: Subroutine name missing.

CODE_51: Subroutine: Missing open bracket.

CODE_52: Subroutine: Missing close bracket.

CODE_54: Subroutine: struct name length is maximum 50 characters, or there is no open and close brackets.

CODE_60: Stack: Stack unsupported stack command.

CODE_66: Expression: Lhs need to be a modifiable value(&) or a memory address(#).

CODE_71: Expression: Lhs is undefined.

CODE_72: Expression: Rhs is undefined.

CODE_76: Undefined behavior: Unknown keyword

CODE_80: Undefined behavior: Wrong syntax.

CODE_81: Undefined behavior: Could not identify the error

CODE_90: Wrong file path.

CODE_91: Can only include a maximum of 100 files in the same project.

CODE_100: Compare: Cannot compare digits and string

CODE_101: Include: cannot include a main file into another main file.

Anmärkning: Att skriva ut minnesadressen för ett värde som inte har en minnes adress på Global kommer föranledas med en krasch det vill säga skriva ut adressen till ett alias som inte existerar. I det fall inget värde finns lagrat på en minnesadress kommer det returnerade värdet vara NULL.

Bilaga Kodexempel

1 Fibonaccis talföljd(loop)

1.1 Exempel 1

```
/* Fibonacci implemented with while loop and Stack */

:sysMemAllocGlobal 128; /* Allocates a Global of size 64 bytes. */

:sysCreateStack 128; /* Create a Stack on the Global of size 32 bytes */


:do {
        :if(:stk.getSize() == 0) {
                /* Push first 2 numbers on Stack. */
```

```
            :stk.pushTop( 0 );
            :stk.pushTop( 1 );

        } :else {
            :stk.pushTop( :stk.getTop() + :stk.getAt( 1 ) ); /* Adds top and second top values  */
        }
} :while(:stk.getSize() < 30);
```

1.2 Exempel 2

```
/* Fibonacci implemented with while loop and alias */

:sysMemAlloc Global 64; /* Allocates a Global of size 64 bytes. */

:sysCreateStack 32; /* Create a Stack on the Global of size 32 bytes */


:alias sum : #32 = 0; /* Sum */

:alias firstNumber : #36 = 0; /* First number */

:alias secondNumber : #40 = 1; /* Second number */

:alias counter : #44 = 0; /* counter */


:print(firstNumber, secondNumber); /* Print first 2 numbers 0, 1 */


/* number of terms to print */

:while(counter != 36) {


        /* sum = first + second */

        &sum = firstNumber + secondNumber;


        /* first = second */

        &firstNumber = secondNumber;


        /* second = sum */

        &secondNumber = sum;
```

```
        /* ++counter */
        &counter = counter +1;

        :print(sum); /* print sum */
};
```

2 Fibonaccis talföljd(rekursion)

2.1 Exempel 1

```
/* Fibonacci implemented with recursion and Stack */
:sysMemAlloc Global 128; /* Allocates 128 bytes size of Global.
:sysCreateStack 128; /* Creates a Stack on Global, with the size 128 bytes, */

:stk.pushTop( 0 ); /* Add first value on Stack */
:stk.pushTop( 1 ); /* Add second value on Stack */
:call fibonacci; /* Call subrutine fibonacci. */

:subroutine fibonacci {
        :stk.pushTop( :stk.getTop() + :stk.getAt( 1 ) ); /* Add top and second top value     from Stack.
        Push the sum of the two values on top of Stack. */
      /* Stopvalue */
        :if( :stk.getSize() < 30 ) {
                :call fibonacci; /* Recursive call */
        }
};
```

2.2 Exempel 2

```
/* Fibinacci implemented with recursion and alias */
:sysMemAlloc Global 64;

:alias sum : #33 = 0; /* Creates an alias on the Global with the name sum */
```

```
:alias firstNumber : #4 = 0; /* Creates an alias on the Global with the name First number */
:alias secondNumber : #8 = 1; /* Creates an alias on the Global with the name Second number */

:print(firstNumber, secondNumber); /* Print first 2 numbers 0, 1 */
:call calcFib;

:subroutine calcFib {
	&sum = firstNumber + secondNumber; /* sum = first + second */
	/* if sum < stop number */
	:if(sum < 300) {
		:print(sum); /* print sum */
		&firstNumber = secondNumber;  /* first = second */
		&secondNumber = sum; /* second = sum */
		:call calcFib;
	}
};
```

Bilaga Testkod

1 Felhantering

```
:sysMemAlloc Global 64;
:sysCreateStack 32;

/*CODE_10*/
/*:alias GlobalOverflow : #64 = "k";*/

/*CODE_11*/
/*:stk.pushAt(32, 12345);*/

/*CODE_12*/
/*:sysCreateStack 65;*/
```

```
/*CODE_13*/
/*:sysCreateStack 0;*/

/*CODE_14*/
/*:sysMemAlloc Global 0;*/

/*CODE_15*/
/*:sysMemAlloc Global 1;*/

/*CODE_16*/
/*:sysCreateStack 1;*/

/*CODE_30*/
/*:alias aliasOnStack : #31 = "crash";*/

/*CODE_31*/
/*:alias s_ : #32 = 12345;*/
/*:alias _a : #32 = 12345;*/
/*:alias 1s : #32 = 12345;*/

/*CODE_32*/
/*:print(aliasName);*/

/*CODE_34*/
/*:alias string #32 = "hej";*/

/*CODE_35*/
/*:alias var : 32;*/
```

```
/*:alias var : 32 = "hej;*/
/*:alias var : #gfd;*/
/*:alias var : #gfd = "hej";*/

/*CODE_3510*/
/*:alias var : #32 = hej;*/
/*:alias var : #32 = 12g4k5;*/
/*:alias var : #32;
&var = 12g4k5;*/

/*CODE_3511*/
/*:alias var : #32 = 9999999999;*/

/*CODE_40*/
/*:struct {
        :alias var : offset(0);
};*/

/*CODE_41*/
/*:struct testStruct
        :alias var : offset(0);
};*/

/*CODE_42*/
/*:struct testStruct {
        :alias var : offset(0);*/

/*CODE_43*/
/*:struct testStruct
```

```
        :alias var : offset(0);*/

/*CODE_50*/
/*:subroutine {
        :alias var : offset(0);
};*/

/*CODE_51*/
/*:subroutine testSub
        :alias var : offset(0);
};*/

/*CODE_52*/
/*:subroutine testSub {
        :alias var : offset(0);*/

/*CODE_53*/
/*:subroutine testSub
        :alias var : offset(0);*/

/*CODE_60*/
/*:alias var : #32 = :stk.get();*/
/*:stk.get();*/

/*CODE_66*/
/*:alias var : #32 = "Hello";
var = 12345;*/

/*CODE_71*/
```

```
 /*= 12345;*/


/*CODE_72*/

/*12345 = ;*/


/*CODE_90*/

/*OK. Lexical.cpp ReadFile(const char*){}*/


/*CODE_90*/

/*:include("C:\Programmering\SimpleInterpreter\syntax\wrongPath.q");*/


/*CODE_91*/

/*OK*/


/*CODE_100*/

/*:if(2 == "2") { :print("ERROR"); } :else { :print("Success"); }*/
```

2 Test if och allias

```
:sysMemAlloc Global 64;

:sysCreateStack 32;


:print("--------------TestIfAndAlias.q--------------");


:alias str : #32 = "hej";

:alias number : #35 = 10 + 5;


:print(number); /*skriver ut värdet 15*/

:print(str); /*skriver ut värdet "hej"*/
```

```
:print(&str); /*Skriver ut addresen 32*/
:print( &number ); /*Skriver ut addresen 35*/

:stk.pushTop(10);
:alias n1 : #39 = :stk.getTop() + 5;
:if(n1 == 15) { :print("Success"); } :else { :print("ERRROR"); }
:print("_____");

:if(#32 +1 == "e") { :print("Success"); } :else { :print("ERROR"); }

/* < > Jämför längderna på textsträngarna eller storleken på heltal*/
:if("n1" < "number") { :print("Success"); } :else { :print("ERROR"); }

:if("100" > "n1") { :print("Success"); } :else { :print("ERROR"); }

:if(n1 < number) { :print("ERROR"); } :else { :print("Success"); }

:if(100 > n1) { :print("Success"); } :else { :print("ERROR");

:if(n1 == number) { :print("Success"); } :else { :print("ERRROR"); }

:if(n1 != number) { :print("ERROR"); } :else { :print("Success"); }

:if("number" == "number") { :print("Success"); } :else { :print("ERRROR"); }

:if("n1" != "number") { :print("Success"); } :else { :print("ERRROR"); }

:if("nej" == "hej") {
        :print("ERROR");
```

```
} :else {
	:if(66 == 88) {
		:print("ERROR");
	} :else {
		:if("success" == "success") {
			:print("success");
		} :else {
			:print("ERROR");
		}
	}
}
:print("_____");

:stk.popTop();
```

3 Loopar

```
:sysMemAllocGlobal 64;
:sysCreateStack 32;

:print("--------------TestLoops.q--------------");

/*While loops*/
:print("thirdWhileLoop");
:stk.pushAt(0, 0);
:while(:stk.getAt(0) != 1) {
		:print("thirdInnerWhileLoop");
		:stk.pushAt(5, 0);
		:while(:stk.getAt(5) != 3) {
			:print(:stk.getAt(5));
```

```
                        :stk.pushAt(5, :stk.getAt(5) +1);
                }
                :print("_____");
        :print(:stk.getAt(0));
        :stk.pushAt(0, :stk.getAt(0) +1);
}
:print("_____");

:print("firstWhileLoop");
:stk.pushAt(0, 0);
:while(:stk.getAt(0) != 5) {
        :print(:stk.getAt(0));
        :stk.pushAt(0, :stk.getAt(0) +1);
}
:print("_____");

:print("secondwhileLoop");
:stk.pushAt(5, 0);
:while(:stk.getAt(5) != 3) {
        :print(:stk.getAt(5));
        :stk.pushAt(5, :stk.getAt(5) +1);
}
:print("_____");


/*Do while loops*/
:print("firstDoWhile");
:stk.pushAt(0, 0);
:do {
```

```
	:print("firstInnerDoWhile");
	:stk.pushAt(5, 0);
	:do {
		:print(:stk.getAt(5));
		:stk.pushAt(5, :stk.getAt(5) +1);

	} :while(:stk.getAt(5) != 3);
	:print("_____");

	:print(:stk.getAt(0));
	:stk.pushAt(0, :stk.getAt(0) +1);

} :while(:stk.getAt(0) != 3);
:print("_____");

:print("secondDoWhile");
:stk.pushAt(0, 0);
:do {
	:print(:stk.getAt(0));
	:stk.pushAt(0, :stk.getAt(0) +1);

} :while(:stk.getAt(0) != 3);
:print("_____");

:print("thirdDoWhile");
:stk.pushAt(5, 0);
:do {
	:print(:stk.getAt(5));
	:stk.pushAt(5, :stk.getAt(5) +1);
```

```
} :while(:stk.getAt(5) != 3);

:print("_____");


/* Erase all elements on Stack */

:stk.popTop();
```

4 Matematiska uttryck

```
:sysMemAllocGlobal 64;

:sysCreateStack 32;


:print("--------------TestMathExpressions.q--------------");


:alias var : #32;

:print(#32 + 1 = 10, "Correct result is 10");


:print(#32 = 1 + 10, "Correct result is 11");


:print(&var = 1000 + 200, "Correct result is 1200");


:print(50 + 40 + 10 - 200, "Correct result is -100");


:print(2 * 4 * 7, "Correct result is 56");


:print(2 * 8 / 2, "Correct result is 8");


:print(2 == 4, "Correct result is false");
```

5 Print

```
:sysMemAllocGlobal 64;

:sysCreateStack 32;


:print("--------------TestPrint.q--------------");

:print("hello world!");

:print("hello  again " , "who are  you?");

:print(""); /*newline*/

:print("g");

:print(" "); /*print empty string*/


:alias test : #33 = 12345;

:print(test);

:print("Correct answer is 12345");

:print("");


:print(2 == 2, "true is Correct");

:print(2 != 2, "false is Correct");


:alias name : #33 = "creater";

:print("My name is: " , creater, " and my place in memory is: " , &name );

:print("Correct answer is: My name is: creater and my place in memory is: 33 ");
```

6 Stack

```
:sysMemAllocGlobal 64;

:sysCreateStack 64;


:stk.pushTop(1);

:stk.pushTop(2);
```

```
:stk.pushTop("hello");
:if( :stk.getTop() == "hello") { :print("Success"); } :else { :print("ERROR"); }

:stk.pop();
:if( :stk.getTop() == "hell") { :print("Success"); } :else { :print("ERROR"); }

:stk.popTop();
:if( :stk.getTop() == 2) { :print("Success"); } :else { :print("ERROR"); }
:print("NULL: ", :stk.getAt(2));

:stk.popTop();
:stk.popTop();
:stk.popTop();
:print("NULL: ", :stk.getTop())


:stk.pushTop(1);
:stk.pushAt(4, "hej");
:stk.pushTop(2);
:if( :stk.getAt(2) == "hej") { :print("Success"); } :else { :print("ERROR"); }

:stk.popTop();
:stk.popTop();
:stk.popTop();
:stk.popTop();


:stk.pushTop(11);
:stk.pushTop(12);
```

```
:stk.pushTop("hello again");

:if( :stk.getTop() == "hello again") { :print("Success"); } :else { :print("ERROR"); }
:if( :stk.getAt(2) == 11) { :print("Success"); } :else { :print("ERROR"); }

:stk.pop();
:if( :stk.getTop() == "hello agai") { :print("Success"); } :else { :print("ERROR"); }

:stk.popTop();
:if( :stk.getTop() == 12) { :print("Success"); } :else { :print("ERROR"); }
:print("NULL: ", :stk.getAt(2));

:stk.popTop();
:stk.popTop();
:stk.popTop();
:print("NULL: ", :stk.getTop());
```

7 Struct

```
:sysMemAllocGlobal 64;
:sysCreateStack 32;

:print("--------------TestStructs.q--------------");

:struct Pair {
        :alias first : offset(0);
        :alias second : offset(9);
};
```

```
:struct Calculator {
	:alias res : offset(0);
	:alias valFirst : offset(4);
	:alias valSecond : offset(8);

	:subroutine add { &res = valFirst + valSecond; };
};

:struct String {
	:alias len : offset(0); /*offset 0*/
	:alias cStr : offset(4); /*offset 4*/
	:alias pPair : Pair : offset(8); /*offset 8*/ /*Transforms to String/pPair/first & String/pPair/second*/

}; /*No typedef needed underlying layer will automaticle create a typedef String*/

/*
:alias string : #33 = String;
&string.pPair.first = "firstPair";
&string.cStr = "hej";
&string.len = 3;

:print(string.pPair);
:print(string.pPair.second);
:print(string.pPair.first);
:print(string);
:print(string.cStr);
:print(string.len);
```

```
:if(string.cStr == "hej") { :print("Success"); } :else { :print("ERROR"); }
:if(string.len == 3) { :print("Success"); } :else { :print("ERROR"); }
:if(string == 3) { :print("Success"); } :else { :print("ERROR"); }
:if(string.pPair == "firstPair") { :print("Success"); } :else { :print("ERROR"); }
:if(string.pPair.second == "secondPair") { :print("Success"); } :else { :print("ERROR"); }
:if(string.pPair.first == "firstPair") { :print("Success"); } :else { :print("ERROR"); }
:print("_____");
*/

:alias calc : #33 = Calculator;
&calc.valFirst = 2;
&calc.valSecond = 3;
:call calc.add;

:print(calc.res);
:print(calc.valFirst);
:print(calc.valSecond);
:print(calc);

:if(calc.res == 5) { :print("Success"); } :else { :print("ERROR"); }
:if(calc.valFirst == 2) { :print("Success"); } :else { :print("ERROR"); }
:if(calc.valSecond == 3) { :print("Success"); } :else { :print("ERROR"); }
:if(calc == 5) { :print("Success"); } :else { :print("ERROR"); }
:print("_____");


:alias pair : #33 = Pair;
&pair.second = "test";
```

```
&pair.first = 67890;


:print(pair.first);
:print(pair.second);
:print(pair);

:if(pair.second == "test") { :print("Success"); } :else { :print("ERROR"); }
:if(pair.first == 67890) { :print("Success"); } :else { :print("ERROR"); }
:if(pair == 67890) { :print("Success"); } :else { :print("ERROR"); }
:print("_____");



:alias pairNext : #43 = Pair;
&pairNext.second = 12345;
&pairNext.first = "testAgain";

:print(pairNext.first);
:print(pairNext.second);
:print(pairNext);



:if(pairNext.first == "testAgain") { :print("Success"); } :else { :print("ERROR"); }
:if(pairNext.second == 12345) { :print("Success"); } :else { :print("ERROR"); }
:if(pairNext == "testAgain") { :print("Success"); } :else { :print("ERROR"); }
:print("_____");
```

8 Subrutiner

```
:sysMemAllocGlobal 64;
:sysCreateStack 32;
```

```
:print("--------------TestSubroutines.q--------------");

:subroutine sub { &sum = val2 - val1; };

:subroutine add { &sum = val1 + val2; };

:subroutine mul { &sum = val2 * val1; };

:subroutine div { &sum = val2 / val1; };

:alias val1 : #43 = 3;
:alias val2 : #47 = 8;
:alias val2Negative : #51 = -8;
:alias sum : #55 = 0;

:subroutine subReverseOrder { &sum = val1 - val2; };

:subroutine addReverseOrder { &sum = val1 + val2; };

:subroutine mulReverseOrder { &sum = val1 * val2; };

:subroutine divReverseOrder { &sum = val1 / val2; };

:call subReverseOrder;
:print("subReverseOrder");
:if(sum == -5) { :print("Success"); } :else { :print("ERROR"); }
:print("_____");
```

```
:call addReverseOrder;
:print("addReverseOrder");
:if(sum == 11) { :print("Success"); } :else { :print("ERROR"); }
:print("_____");

:call mulReverseOrder;
:print("mulReverseOrder");
:if(sum == 24) { :print("Success"); } :else { :print("ERROR"); }
:print("_____");

:call divReverseOrder;
:print("divReverseOrder");
:if(sum == 0) { :print("Success"); } :else { :print("ERROR"); }
:print("_____");

:call sub;
:print("sub: ");
:if(sum == 5) { :print("Success"); } :else { :print("ERROR"); }
:if(val2 == 8) { :print("Success"); } :else { :print("ERROR"); }
:if(val1 == 3) { :print("Success"); } :else { :print("ERROR"); }
:print("_____");

:call add;
:print("add: ");
:if(sum == 11) { :print("Success"); } :else { :print("ERROR"); }
```

```
:if(val2 == 8) { :print("Success"); } :else { :print("ERROR"); }
:if(val1 == 3) { :print("Success"); }  :else { :print("ERROR"); }
:print("_____");

:call mul;
:print("mul: ");
:if(sum == 24) { :print("Success"); }  :else { :print("ERROR"); }
:if(val2 == 8) { :print("Success"); } :else { :print("ERROR"); }
:if(val1 == 3) { :print("Success"); }  :else { :print("ERROR"); }
:print("_____");

:call div;
:print("div: ");
:if(sum == 2) { :print("Success"); } :else { :print("ERROR"); }
:if(val2 == 8) { :print("Success"); } :else { :print("ERROR"); }
:if(val1 == 3) { :print("Success"); } :else { :print("ERROR"); }
:print("_____");



:subroutine subNegative { &sum = val1 - val2Negative; };

:subroutine addNegative { &sum = val1 + val2Negative; };

:subroutine divNegative { &sum = val1 / val2Negative; };

:subroutine mulNegative { &sum = val1 * val2Negative; };
```

```
:subroutine nestedSubroutineCall {
                    :call sub;
                    :stk.pushTop(sum);
                    :call add;
                    &sum = sum + :stk.getTop(); /*5 + 11 = 16*/
};

:print("nestedSubroutineCall: ");
:call nestedSubroutineCall;

:if(sum == 16) { :print("Success"); } :else { :print("ERROR"); }
:stk.popTop();
:print("_____");

:call subNegative;
:print("subNegative");
          /* 3 - -8 */
:if(#55 == 11) { :print("Success"); } :else { :print("ERROR"); }
:print("_____");


:call addNegative;
:print("addNegative");
:if(#55 == -5) { :print("Success"); } :else { :print("ERROR"); }
:print("_____");


:call divNegative;
:print("divNegative");
```

```
:if(#55 == 0) { :print("Success"); } :else { :print("ERROR"); }

:print("_____");



:call mulNegative;

:print("mulNegative");

:if(#55 == -24) { :print("Success"); } :else { :print("ERROR"); }

:print("_____
```

Bilaga Övningar

Övningar ett till och med sju är tänkta att implementeras med programmeringsspråket Q resterande övningar i programmeringsspråket C.

1

Create a "stack overflow"(use recursion).

Solution:

```
/*assignment1.main.q*/

    :sysMemAllocGlobal 64; /* Allocates a Global of size 64 bytes. */

    :sysCreateStack 32; /* Create a Stack on the Global of size 32 bytes */


    :call stackoverflow;

    :subroutine stackoverflow {

        :stk.pushTop( 1 ); /* Push value to top of Stack. */

        :call stackoverflow; /* Calling the subroutine again, this is recursion! */

        /* The recursion does not have a stop statement, a new value will be
        pushed until Stack runs out of memory (#33) and crash. */

    };
```

2

Create a "buffer overflow"(use loop).

Solution:

```
/*assignment2.main.q*/
        :sysMemAllocGlobal 32; /* Allocates a Global of size 32 bytes. */
        :alias str : #0 = "Hello World";
        :do {
                :&str = str + "again "; /* adds 'again' to alias str */

} :while(#33 != "a");  /* This will never stop loop because buff overflow will happen on line before store value on #32 */
```

3

Create a subroutine that adds two numbers(subroutine shall only contain calculation). Print the result value and address outside the subroutine.

Solution:

```
/*assignment3.main.q*/
:sysMemAllocGlobal 64; /* Allocates a Global of size 64 bytes. */
:sysCreateStack 32; /* Create a Stack on the Global of size 32 bytes */

:alias sum : #33; /* Creates an alias on address #33. */
:stk.pushTop( 2 ); /* Push number 2 on Stack */
:stk.pushTop( 3 );

:call add; /* Calls subroutine add */
:print( sum ); /* Prints alias sum value to console window. */
:print( &sum ); /* Prints alias sum address to console window. */

:subroutine add {
        &sum = :stk.getTop() + :stk.getAt( 1 ); /* Adds number 3 and 2.  */
};
:stk.popTop(); /* Clean Stack. */
:stk.popTop();
```

4

Simulate a function that calculate the length of a string(strlen), use the stack.

Solution:

```
/*assignment4.main.q*/

:sysMemAllocGlobal 64; /* Allocates 64 bytes */

:sysCreateStack 32; /* Creates a 32 bytes Stack on the Global. */

:alias str : #36 = "this is a test string"; /* Creates alias str on Global location #36 */

:alias len : #32 = 0; /* Creates alias len on Global location #32. */

:call strlen; /* Call subroutine strlen. */


:subroutine strlen {

        :stk.pushTop(len); /* Push alias len value to the top of the Stack. */

        :stk.pushTop(str);


        :while(:stk.getSize() == 2) {

                :stk.pushAt(1, :stk.getAt(1) +1); /* Increment the string length counter by
1. */

                :stk.pop(); /* Erase one letter from "this is a test string" until there is no
                letters left, size of Stack will be one and leave while loop.  */

        }

        &len = :stk.getTop(); /* Moves value on top of Stack to alias len */

};
```

5

Print the value and address from an undefined reference, don't forget to write and briefly explain the results from the printing statements.

Solution:

```
/*assignment5.main.q*/

        :sysMemAllocGlobal 64;

        :alias val : #32;

        :print( val ); /* alias points to address #32 */
```

```
:print( &val ); /* alias reference to what is storred inside #32 in this case that is
undefined but since we only overwrites memory instead of clean/free there is a
possibility that the memory address could contain old data. */
```

6

Create a struct myStruct that contains a string and a lenght for that string, create an alias to myStruct. Create a subroutine that sets the values inside myStruct (lenght shall be >= 6). Print the value of the alias, print second letter of the string and print the address to the second letter of the string. dont forgett to write and briefly explain the results from the printing statements.

Solution:

```
/*assignment6.main.q*/

        :sysMemAllocGlobal 64;

        :struct myStruct {

                :alias len : offset( 0 ); /* 4 bytes */

                :alias str : offset( 4 ); /* only limited by Global size. */

        };


        :alias pStruct : #32 = myStruct; /* alias to myStruct. */

        :call init;


        :print( #32 ); /* it will print number 11. */

        :print( pStruct.str + 5 ) /* it will print 'o'. */

        :print( & pStruct + 5 ) /* it will print #37  */


         :subroutine init {

                & pStruct.str = "hello world";

                & pStruct.len = 11;

        }
```

7

Assignment 7:

Print the address from an NULL pointer, don't forget to write and briefly explain the results from the printing statement.

Solution:

```
//main.c

#include <stdlib.h>

int main() {
	int *p = NULL;
	printf("%p\n", (void*) &p);
	system("pause");
	return 0;
}
```

8

Assignment 8:

Implement a function that add two numbers and describe what is happening in the registers.

Solution:

```
//main.c

#include <stdlib.h>

int main() {
	add(2, 3);


	system("pause");
	return 0;
}


int add(int v1, int v2) {
	/*__asm {
		mov eax, v1;
		mov ebx, v2;
```

```
		add eax, ebx;
	}*/
	return v1 + v2;
}
```

9

Assignment 9:

Implemet a function that calculates the lenght of a char pointer. Think in terms of pointers and not arrays(forbbiden to use [] instead use *pointer).

Solution:

```
//main.c
#include <stdlib.h>
int main() {
	char *cStr = "hello\0";
	printf("%i\n", strlen(cStr));

	system("pause");
	return 0;
}

int strlen(char *cStr) {
	size_t len = 0;

	while (*(cStr + len) != '\0') {
		++len;
	}
	return len;
}
```

10

Assignment 10:

Create 2 NULL pointers(int), Save 5 in the address the first pointer is pointing to, print the newly inserted value. Let the second pointer point to the address of the first pointer, print the second pointer. Briefly describe and try to think in terms of pointers and not arrays(forbbiden to use [] instead use *).

Solution:

```
//main.c

#include <stdlib.h>

int main() {
	int *p1 = NULL;
	int *p2 = NULL;
	p1 = 5; //saves 5 to the address p1 is pointing to.
	printf("%i\n", p1);
	p2 = &p1; //p2 points to the first memory address of p1.
	printf("%i\n", *(p2 + 0)); //dereferencing the pointer to get the value.
	system("pause");
	return 0;
}
```