Opgave 1

```
//calculating sin(x) with the precision of n terms

// Function to calculate power (x^y)
double power(double base, double exponent) {
    double result = 1.0;
    for (int i = 0; i < exponent; i++) {
        result *= base;
    }
    return result;
}

// Function to calculate factorial (n!)
double factorial(int num) {
    double result = 1.0;
    for (int i = 2; i <= num; i++) {
        result *= i;
    }
    return result;
}

double taylor_sine(double x, int n) {

double sinus = 0.0;

    for(int i = 0; i < n; i++) { //counting up to precision of n
        int n_1 = 2*i + 1; //calculating the exponent (the current value of n in the loop)

        double term = (power(x,n_1)/factorial(n_1)); //

        if(i % 2 == 1){     //if the current index in the taylor series is uneven 1,3,5,... we suptract it
          sinus -= term;
        }
        else {              //if the current index in the taylor series is even 0,2,4,... we add it
          sinus += term;
        }
    }

  return sinus;
}
```

```c
//x=0
    double a = sin(0);
    printf("%lf ", a);

    double b = taylor_sine(0, 8);
    printf("%lf \n", b);
    // the output is identical

//x=-5
    double c = sin(-5);
    printf("%lf ", c);

    double d = taylor_sine((-5), 8);
    printf("%lf \n", d);
    // the output close to eachother

//x=10000
    double e = sin(10000);
    printf("%lf ", e);

    double f = taylor_sine(10000, 8);
    printf("%lf \n", f);
    // the output is not near eachother. the high value of x and low value of n makes the taylorfunction not precise

//x=8927398
    double g = sin(8927398);
    printf("%lf ", g);

    double h = taylor_sine(8927398, 8);
    printf("%lf \n", h);
    // the output is even further from eachother. the greater the value of x -> the less precise output

    return 0;
}
```

```
0.000000 -0.000001
0.000000 0.000000
0.958924 0.960921
-0.305614 -7647147672801030501802020095175985313843217694 72.000000
0.958924 0.960921
-0.305614 -7647147672801030501802020095175985313843217694 72.000000
-0.129376 -139435655214478332026692139316195365682359643775220998314540281923894035374144539818842914816.000000
```

Den er dårlig til høje tal, men små (både positive of negative) x værdier er okay.

Opgave 2

```c
#include "stack.h"
#include <assert.h>
#include <stdlib.h>

void initialize(stack *s) {
  assert(s != 0);

  node *p = s->head; //starting with the head (first element)

    while (p->next != NULL) {
      p = p->next;
    } // p points to the last element

    s->head = NULL; //setting the size to 0
}


void push(int x, stack *s) {
  assert (s != 0);

  node *q_push = (node *)malloc(sizeof(node)); //adding a new node and updating the size of the list

  q_push->data = x; //giving the new node 'q' the value of x
  q_push->next = s->head; //setting 'q' as the new head
  s->head = q_push; //updating the stack, so the node 'q' is on top
}

int pop(stack *s) {
  assert (s != 0);

  int q_pop = s->head->data; //take the top value

  node *temporary = s->head; //make a placeholder for the top value
  s->head = s->head->next; // update the head, so it moves to the next element

  free(temporary); //letting go of the old head (freeing the top element)
  return q_pop; //returning the top value, that we poped
}

bool empty(stack *s) {
//returns false if the stack is NOT empty. returns true if the stack IS empty
  return (s->head == NULL);
}

bool full(stack *s) {

  node *fiction = (node *)malloc(sizeof(node)); //making a fictional node to check for space

  if (fiction == NULL) { //NULL meaning there is no more space = return true = the stack is full
        return true;
  }

  free(fiction); // else freeing the fictional note and returning false = the stack is NOT full

  return false;
}
```

Jeg kan ikke få lov at pushe min kode fordi githuben er puplic og den ikke vil give den min private mail 😊