TENTAMEN

DIT632-DIT633 march 2024

Kurskod	--
Bedömningsform	--
Starttid	12.03.2024 14:00
Sluttid	12.03.2024 18:00
Bedömningsfrist	--
PDF skapad	04.06.2024 11:04
Skapad av	Helny Malmborg

ℹ Instructions for the exam

DIT 633 - Development of Embedded and Real-time systems

This exam should be an individual work for you. You are not allowed to use any outside help.

If you are allowed to use a compiler, there is a link to an online one, which will open in a separate window. You can test the code in the online compiler, but **you must remember to copy-paste it back to the exam**, otherwise your code will disappear once you close the window.

The same is true for TinkerCad, please remember to copy-paste the code from TinkerCad to the exam.

You are NOT allowed to access code that was not written during the exam (e.g., from your previous assignments). This will be considered plagiarism.

You are not allowed to copy code from your colleagues or any other external source.

Remember: In programming questions, if the code does not compile, you get 0 points for the question!

Grading scale:
50% correct - 3
65% correct - 4
85% correct - 5

Good luck!

/Miroslaw
031 772 1081

1 Sustainability

Explain the concept of **Embodied carbon**. Provide an example of how one can work to minimize the effects of it.

Grading: 2 points

Skriv in ditt svar här

This concept is related to the sustainability efforts regarding the reduction of carbon emissions related to software and, in particular, to the emissions that are directly associated to embedded systems.

For example, even if certain equipments are not emitting carbon themselves, and instead use "clean" sources of, such as electricity, they still are responsible for a considerable amount of carbon release since they were possibly built resorting to carbon, or the electricity they use has been produced through processes that release carbon.

To minimize the effects of this, it's important to not mass-produce equipments with short lifespan, make sure that software and hardware are made with compatibility in mind in order to not render equipments useless in a short frame. It is also important to have software and hardware that run in a lean way, as in, not spending too much energy and that can run without requiring a huge carbon effort to build. It is also worth considering the use of energy that is produced in a cleaner way, for example by dynamically shifting the execution of heavy functions to places on earth where electricity is obtained in greener ways.

Ord: 191

Besvarad.

2 Pointers

When working with embedded software, we often use pointers to make the memory handling more efficient and to reduce the carbon footprint of our systems.

Here, you need to correctly recognize what the pointers are.

Please choose the right interpretation of **x** in each of the statements:

int *x();

- x is a pointer to a function that returns an int
- x is a pointer to a variable of type int
- [✓ x is a function that returns a pointer to an int] ✓
- x is a variable of type int

int (*x)();

- x is a function that returns a points to an int
- x is a pointer to a funtion that takes a pointer as an argument
- [✓ x is a pointer to a function that returns an int] ✓
- x is a function that returns an int

int * (* (* * x [ ] []) ( )) [8];

- x is a function that takes as input an array of 8 pointers to pointer to an array and returns a pointer to array of pointer to int
- x is an array of 8 pointers to functions, where each function returns a pointer to a fun[illegible]on returning a pointer to a function returning a pointer to an integer ✓
- [✓ x is array of array of 8 pointers to functions returning pointer to array of pointer to a p[illegible]ter to int] ✗
- x is an array of pointers to pointer to function returning 8 pointers to array of pointer to int

int (* (* x) [ ]) ( );

- ○ x is a function that takes as an argument a pointer to an array of pointers and returns a pointer to int
- ○ x is a function that takes as argument an array of pointers to functions and return a pointer to int
- ○ x is an array of pointers to functions that take no arguments and return pointers to int
- [✓ x is a pointer to an array of pointers to functions returning an int] ✓

int (* x ( )) [20];

- [✓ x is a function returning a pointer to an array of 20 elements of type int] ✓
- ○ x is an array of 20 pointers to functions returning int
- ○ x is an array of pointers to functions returning pointers to functions returning pointers to int
- ○ x is a pointer to a function returning a pointer to an array of 20 elements of type int

Delvis rätt. 4 av 5 poäng.

3 Bitpacking for Easter bunny and Jack Frost

Easter bunny and Jack Frost compete for who gets to decide how Easter is supposed to look like. Easter bunny wants the Easter to be very much like the Spring should be, with flowers and sun. Jack Frost wants the Easter celebrations to be another white holidays with snow and frost in the windows.

In 2024, they decided that they will settle the score once and for all by using a program in C.

However, they did not read DIT633 and they need your help. So, they came up with the following game.

Each of them gets to draw 1 byte ten times. Based on the content of that byte, they make a move. The one that moves the furthest in 10 draws, wins.

Here is the content of the byte as they agreed on:

Bit 0 (MSB) - who gets to move:

- 1 - bunny gets the move
- 0 - Jack gets to move

Bit 1 - direction:

- 1 - move forward
- 0 - move backward

Bits 2-3 - speed multiplier:

- the muiltiplier for the number of steps (bits 5-8)

Bits 4-7 - steps:

- the number of steps that they move in this round.

For example, byte 0xD1 would result in this bit assignment:

bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
1	1	0	1	0	0	0	1

This means that the Easter bunny gets to move 1 step forward.

Your task:
Write a program that will simulate this game. The program should randomly draw the bytes for both the Easter bunny and Jack Frost. It should print the status for each round and it should print in the end who wins.

An example printout:
Welcome to the game!
Bunny draws 1 byte: 0xD1
Bunny moves one step forward, bunny's position: 1, Jack's position: 0
Jack draws 1 byte: 0xD2
Bunny moves one step forward, bunny's position: 2, Jack's position: 0

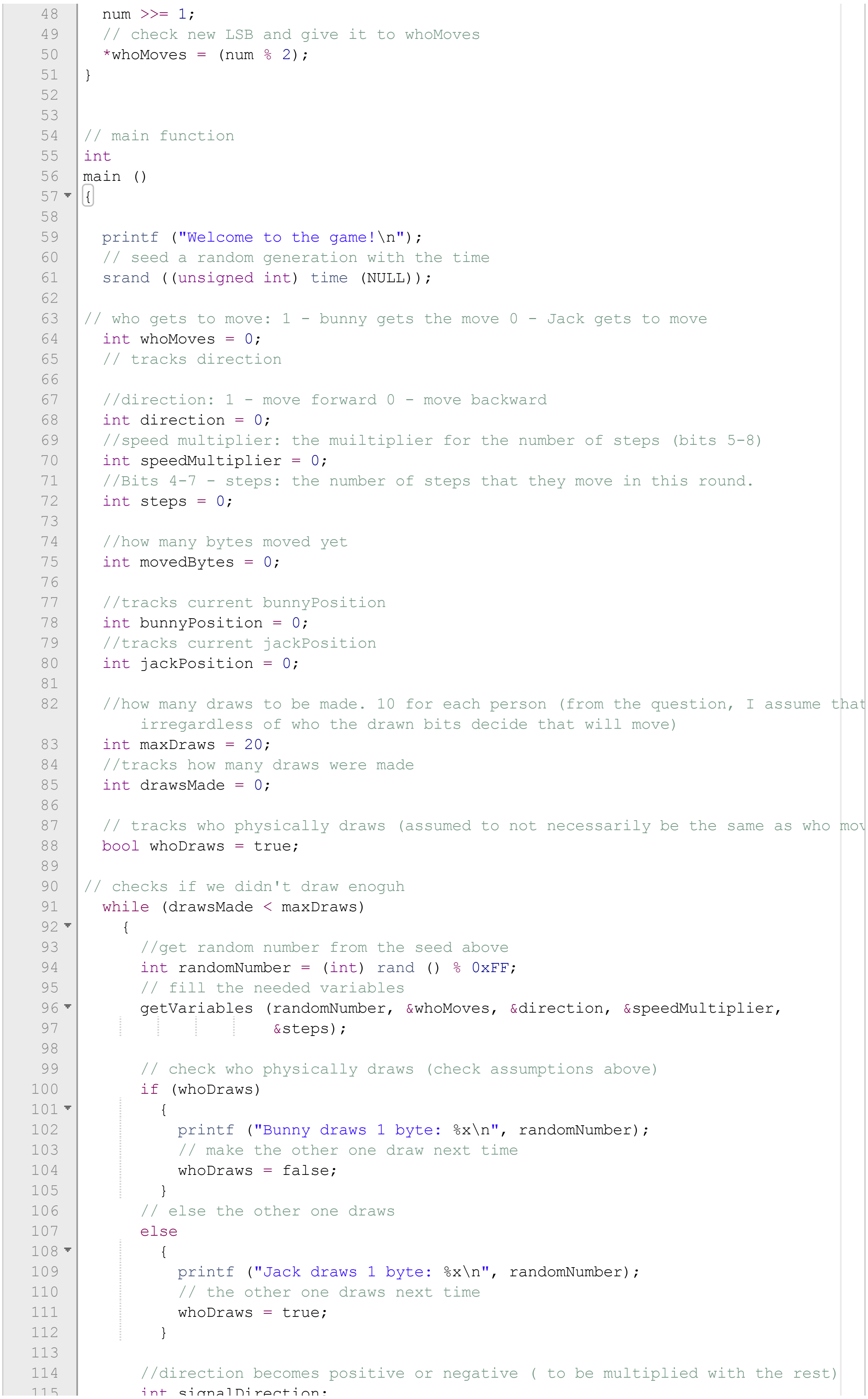
Bunny draws 1 byte: 0x51
Jack moves one step forward, bunny's position: 2, Jack's position: 1
...
...
Game ends: <Easter bunny / Jack Frost> wins! Bunny's position: <XXX>, Jack's position: <YYY>

Grading:
1. Correct implementation of the game: 5 points
2. Using bit operations: 3 points
3. Commenting the code: 2 points

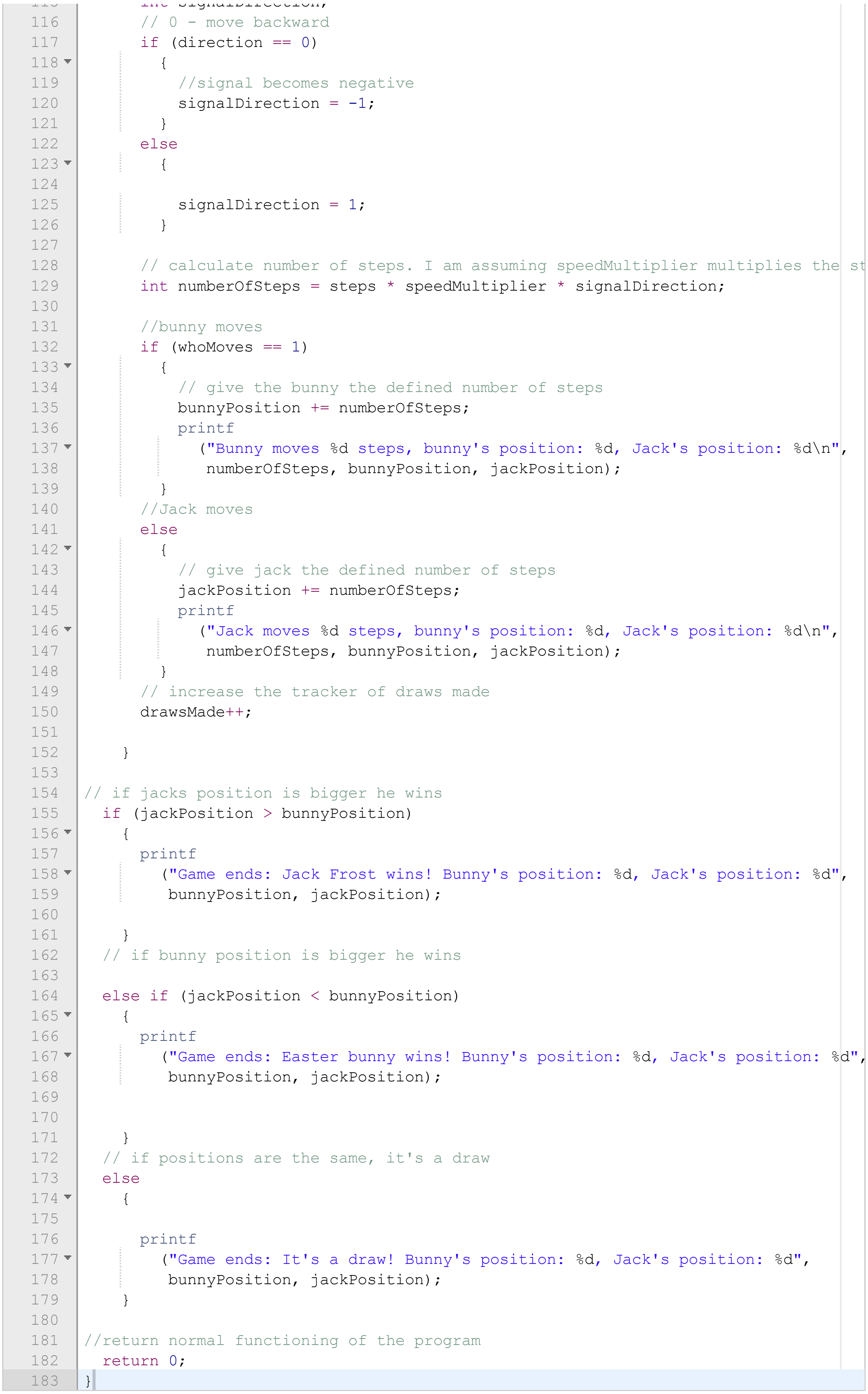
In this question, you can use the online compiler here

Skriv in ditt svar här

```c
//library with standard input output functions
#include <stdio.h>
//library with standard functions
#include <stdlib.h>
//library with standard time functions
#include <time.h>
//library with support for booleans
#include <stdbool.h>
//library with support for types
#include <ctype.h>
// library with standard string functions
#include <string.h>

// function to analyse the bits of the provided number and give them to the appropr
void
getVariables (unsigned int num, int *whoMoves, int *direction,
              int *speedMultiplier, int *steps)
{
// check LSB and give it to steps
  *steps = num % 2;
  // discard bit (already got the info we needed)
  num >>= 1;

  // check new LSB and give it to steps with proper shift
  *steps += ((num % 2) << 1);
  // discard bit (already got the info we needed)
  num >>= 1;

  // check new LSB and give it to steps with proper shift
  *steps += ((num % 2) << 2);
  // discard bit (already got the info we needed)
  num >>= 1;
  // check new LSB and give it to steps with proper shift
  *steps += ((num % 2) << 3);
  // discard bit (already got the info we needed)
  num >>= 1;
  // check new LSB and give it to speedMultiplier
  *speedMultiplier = (num % 2);
  // discard bit (already got the info we needed)
  num >>= 1;
  // check new LSB and give it to speedMultiplier with proper shift
  *speedMultiplier += ((num % 2) << 1);
  // discard bit (already got the info we needed)
  num >>= 1;
  // check new LSB and give it to direction
  *direction = (num % 2);
  // discard bit (already got the info we needed)
```



```c
  num >>= 1;
  // check new LSB and give it to whoMoves
  *whoMoves = (num % 2);
}


// main function
int
main ()
{

  printf ("Welcome to the game!\n");
  // seed a random generation with the time
  srand ((unsigned int) time (NULL));

// who gets to move: 1 - bunny gets the move 0 - Jack gets to move
  int whoMoves = 0;
  // tracks direction

  //direction: 1 - move forward 0 - move backward
  int direction = 0;
  //speed multiplier: the muiltiplier for the number of steps (bits 5-8)
  int speedMultiplier = 0;
  //Bits 4-7 - steps: the number of steps that they move in this round.
  int steps = 0;

  //how many bytes moved yet
  int movedBytes = 0;

  //tracks current bunnyPosition
  int bunnyPosition = 0;
  //tracks current jackPosition
  int jackPosition = 0;

  //how many draws to be made. 10 for each person (from the question, I assume that
      irregardless of who the drawn bits decide that will move)
  int maxDraws = 20;
  //tracks how many draws were made
  int drawsMade = 0;

  // tracks who physically draws (assumed to not necessarily be the same as who mov
  bool whoDraws = true;

// checks if we didn't draw enoguh
  while (drawsMade < maxDraws)
    {
      //get random number from the seed above
      int randomNumber = (int) rand () % 0xFF;
      // fill the needed variables
      getVariables (randomNumber, &whoMoves, &direction, &speedMultiplier,
		    &steps);

      // check who physically draws (check assumptions above)
      if (whoDraws)
	{
	  printf ("Bunny draws 1 byte: %x\n", randomNumber);
	  // make the other one draw next time
	  whoDraws = false;
	}
      // else the other one draws
      else
	{
	  printf ("Jack draws 1 byte: %x\n", randomNumber);
	  // the other one draws next time
	  whoDraws = true;
	}

      //direction becomes positive or negative ( to be multiplied with the rest)
      int signalDirection;
```



```
      int signalDirection;
      // 0 - move backward
      if (direction == 0)
        {
          //signal becomes negative
          signalDirection = -1;
        }
      else
        {

          signalDirection = 1;
        }

      // calculate number of steps. I am assuming speedMultiplier multiplies the st
      int numberOfSteps = steps * speedMultiplier * signalDirection;

      //bunny moves
      if (whoMoves == 1)
        {
          // give the bunny the defined number of steps
          bunnyPosition += numberOfSteps;
          printf
            ("Bunny moves %d steps, bunny's position: %d, Jack's position: %d\n",
             numberOfSteps, bunnyPosition, jackPosition);
        }
      //Jack moves
      else
        {
          // give jack the defined number of steps
          jackPosition += numberOfSteps;
          printf
            ("Jack moves %d steps, bunny's position: %d, Jack's position: %d\n",
             numberOfSteps, bunnyPosition, jackPosition);
        }
      // increase the tracker of draws made
      drawsMade++;

    }

// if jacks position is bigger he wins
  if (jackPosition > bunnyPosition)
    {
      printf
        ("Game ends: Jack Frost wins! Bunny's position: %d, Jack's position: %d",
         bunnyPosition, jackPosition);

    }
  // if bunny position is bigger he wins

  else if (jackPosition < bunnyPosition)
    {
      printf
        ("Game ends: Easter bunny wins! Bunny's position: %d, Jack's position: %d",
         bunnyPosition, jackPosition);


    }
  // if positions are the same, it's a draw
  else
    {

      printf
        ("Game ends: It's a draw! Bunny's position: %d, Jack's position: %d",
         bunnyPosition, jackPosition);
    }

//return normal functioning of the program
  return 0;
}
```

Besvarad.

4 Validation

During our visit to Volvo, we got to see an example of a problem faced during the validation of software systems.

As a reminder: Pre-requisite: Each card have two sides. One side is a letter and the other is a digit.

Their requirement: If the letter is A then the digit must be 3, if the letter is B then the digit must be 4
Their task: check if the cards K-5 and A-3 fulfill the requirement.

Every time the requirement changes, they need to re-write the program for validation of cards. We can do better than that.

Your task
Write a program that takes the requirements as command line arguments, then randomly generates 10 cards, checks if the cards fulfill the requirements. Then it asks the user for one card and checks if it fulfills the requirement. The program should be able to take 1-10 requirements as input in the command line.

For example
Input (command line): *main.exe A-3 B-2 D-0*

Output:
Requirements:
A-3
B-2
D-0
----
generating: A-3 -- OK
generating: C-0 -- OK
generating: D-2 -- not OK
...

Please input a card: <D-0>
Card <D-0> -- OK

Grading:
1. Correct funtionality - 4 points
2. Using pointers and dynamic arrays - 2 points
3. Commenting the code - 2 points
4. Fail-safety - 2 points

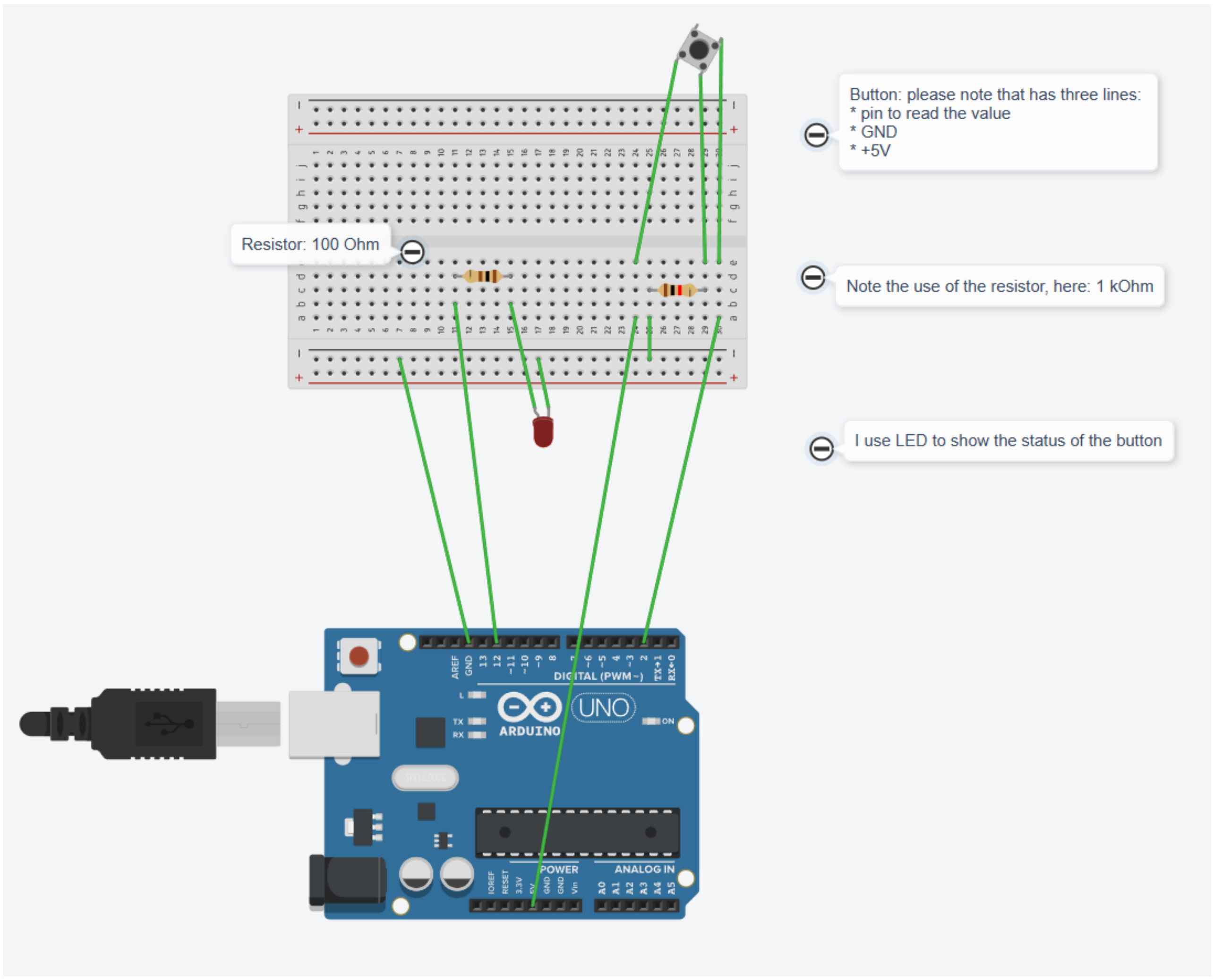
You can use the online compiler in this question here

Skriv in ditt svar här

```c
//library with standard input output functions
#include <stdio.h>
//library with standard functions
#include <stdlib.h>
//library with standard time functions
#include <time.h>
//library with support for booleans
#include <stdbool.h>
//library with support for types
#include <ctype.h>
// library with standard string functions
#include <string.h>

// function that checks cards against requirements
int
main (int argc, char *argv[])
{

  // seed a random generation with the time
  srand ((unsigned int) time (NULL));

  // check if no args provided
  if (argc < 2)
    {
      printf
        ("Provide requirements as arguments (up to 10), this way: main.exe A-3 B-2
      //return with error
      return -1;
    }
  // check if too many args provided
  else if (argc > 11)
    {
      printf
        ("You provided too many arguments. Provide requirements as arguments (up to
            );
      // return with error
      return -1;
    }

  printf ("Requirements:\n");

  // loop through the printing lines of the requirements
  for (int i = 1; i < argc; i++)
    {
      //check if valid letter
      if (argv[i][0] > 90 || argv[i][0] < 65)
        {
          printf
            ("Make sure to provide a valid letter (caps lock letters only) %c",
             argv[i][0]);
          return -1;
        }
      //check if valid number
      if (argv[i][2] < 47 || argv[i][2] > 60)
        {
          printf ("Make sure to provide a valid number %c %d", argv[i][2],
                  argv[i][2]);
          return -1;
        }
      // print letter character and number character from each argument
      printf ("%c-%c\n", argv[i][0], argv[i][2]);

    };

// loop through the 10 card generations
  for (int j = 0; j < 10; j++)
    {
      //generate a random letter cast to char
      char randomChar = (char) (rand () % 25) + 65;
      char randomChar = (char) (rand () % 25) + 65;
      // generate a random number cast to char
      char randomNumber = (char) (rand () % 9) + 49;

      // track when no requirement violations have been detected
      bool ok = true;
      // loop through each requirement, present in the args
      for (int i = 1; i < argc; i++)
        {
          // check if the random letter matches one from a requirement
          if (argv[i][0] == randomChar)
            {
              // check if a random number matches one from a requirement - when a l
              if (argv[i][2] == (randomNumber))
                {
                  // if both match, no violation
                }
              else
                {
                  // if letter matches, but not number, this is a violation - ok be
                  ok = false;
                }
            }
          else
            {
              // if none match, no violation
            }


        };
      // if no violation detected
      if (ok)
        {
          printf ("generating: %c-%d -- OK\n", randomChar,
                  (int) randomNumber - 48);

        }
      // else, a violation was detected
      else
        {
          printf ("generating: %c-%d -- not OK\n", randomChar,
                  (int) randomNumber - 48);

        }
    }

  // char to receive another card
  char more[4];
  // function to receive more a card from the input
  fgets (more, sizeof (more), stdin);
  // the new letter received
  char moreLetter = more[0];
  // the new number received
  char moreNumber = more[2];

  //check if valid letter
  if (moreLetter > 90 || moreLetter < 65)
    {
      printf
        ("Make sure to provide a valid letter (caps lock letters only) %c",
         moreLetter);
      return -1;
    }
  //check if valid number
  if (moreNumber < 47 || moreNumber > 60)
    {
      printf ("Make sure to provide a valid number %c %d", moreNumber,
              moreNumber);
      return -1;
    }


  // track when no requirement violations have been detected
  bool ok = true;
  // loop through each requirement, present in the args
  for (int i = 1; i < argc; i++)
    {
      // check if the random letter matches one from a requirement
      if (argv[i][0] == moreLetter)
        {
          // check if a random number matches one from a requirement - when a lette
          if (argv[i][2] == (moreNumber))
            {
              // if both match, no violation
            }
          else
            {
              // if letter matches, but not number, this is a violation - ok become
              ok = false;
            }
        }
      else
        {
          // if none match, no violation
        }


    };
  // if no violation detected
  if (ok)
    {
      printf ("generating: %c-%d -- OK\n", moreLetter, (int) moreNumber - 48);

    }
  // else, a violation was detected
  else
    {
      printf ("generating: %c-%d -- not OK\n", moreLetter,
              (int) moreNumber - 48);

    }

// program return with no errors
  return 0;

}
```

Besvarad.

5 Blinkers...


Button: please note that has three lines:
* pin to read the value
* GND
* +5V
Resistor: 100 Ohm
Note the use of the resistor, here: 1 kOhm
I use LED to show the status of the button
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN

One of the vehicle manufacturers in Gothenburg has a problem with the blinkers on their vehicles. Whatever they do, they cannot get them to work properly. They've heard about DIT633 and they asked for your help.

Your task

Write a program that will control right-hand side blinkers, based on the circuit above.

When the button is pressed, it starts the interrupt on timers, which makes the blinkers blink. Blinking means that the LED is turned on for 300 miliseconds every 2 seconds, until the button is pressed again.

You will need the code to set-up the interrupt for the timer, which is here (NOTE! - this code sets up the time for 1 second, not 2):

```
void setupTimer1sec() {
  noInterrupts();
  // Clear registers
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1 = 0;
  // 1 Hz (16000000/((15624+1)*1024))
  OCR1A = 15624;
  // CTC
  TCCR1B |= (1 << WGM12);
  // Prescaler 1024
  TCCR1B |= (1 << CS12) | (1 << CS10);
  // Output Compare Match A Interrupt Enable
  TIMSK1 |= (1 << OCIE1A);
  interrupts();
}
```

The signature of the interrupt handler is like this:

```
ISR(TIMER1_COMPA_vect) {
  // TODO: write the necessary functionality here
}
```

Grading:
1. Correct implementation of the functionality - 4 points
2. Using interrupts for the button - 3 points
3. Using interrupts for the timer - 3 points

In this question you are allowed to use TinkerCad [here]()

Skriv in ditt svar här

```
// two seconds timer interrupt
void setupTimer2sec() {
  noInterrupts();
  // Clear registers
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1 = 0;
  // 1 Hz (16000000/((15624+1)*1024))
  // make it trigger every two seconds
  OCR1A = 15624*2;
  // CTC
  TCCR1B |= (1 << WGM12);
  // Prescaler 1024
  TCCR1B |= (1 << CS12) | (1 << CS10);
  // Output Compare Match A Interrupt Enable
  TIMSK1 |= (1 << OCIE1A);
  interrupts();
}
// initialize buttonSensor
int buttonSensor = 2;
// initialize led
int led = 12;
// tracking counter (for the button)
int counter = 0;
//ISR for the two second interrupt
ISR(TIMER1_COMPA_vect) {
  // check if button is in position to be triggered
  if (counter % 4!=0){
    // if so light it up
  digitalWrite(led, HIGH);
    // wait 300 ms
  delay(300);
    // put it down
    digitalWrite(led, LOW);}
}
// my own interrupt for the counter of the button
// light only lights if the counter is at certain numbers
// clicking a button triggers this twice each time
void myISR()
{
  // increase counter
  counter++;
}

// setup of the arduino
void setup()
{
  // assign button sensor the input mode
  pinMode(buttonSensor, INPUT);
  // assign led the output mode
  pinMode(led, OUTPUT);
  // setup the timer
  setupTimer2sec();
  // setup the interrupt for the button
  attachInterrupt(digitalPinToInterrupt(buttonSensor),
                  myISR,
                  CHANGE);
  // begin monitoring
  Serial.begin(9600);
}
// loop of the arduino - nothing happening
void loop()
{
}
```

Besvarad.

6 Handling multiple hardware'

The following source code does not compile, because it contains two variants in the same code. Use the mechanisms to adjust the compilation process to use only one of the variants. You are NOT allowed to change the body or signature of any function.

The body of the main function is missing too. It should ask the user for the number of elements in the array, then it should randomly fill the array with numbers. Then it should print the array before and after the sorting.

```
#include <stdio.h>
// YOU CANNOT MODIFY THE CODE BELOW
void sort(int arr[], int n) {
   int i, j, temp;
   for (i = 0; i < n-1; i++) {
      for (j = 0; j < n-i-1; j++) {
         if (arr[j] > arr[j+1]) {
            // Swap arr[j] and arr[j+1]
            temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
         }
      }
   }
}
// YOU ARE NOT ALLOWED TO MODIFY THE CODE ABOVE

// YOU ARE NOT ALLOWED TO MODIFY THE CODE BELOW
void sort(int arr[], int n) {
   int i, key, j;
   for (i = 1; i < n; i++) {
      key = arr[i];
      j = i - 1;
      /* Move elements of arr[0..i-1], that are
        greater than key, to one position ahead
        of their current position */
      while (j >= 0 && arr[j] > key) {
         arr[j + 1] = arr[j];
         j = j - 1;
      }
      arr[j + 1] = key;
   }
}
// YOU ARE NOT ALLOWED TO MODIFY THE CODE ABOVE

int main() {
  // add your test code here
   return 0;
}
```

Your task

1. Fix the program to compile

2. Add the body of the main() function to test the solution
3. Provide a description (in the comment) how to use your solution

Tips! We used such mechanisms to handle different operating systems and different version of the pump software (guest lecture).

Grading
1. Fixing the program - 2 points
2. Adding body of the main() - 2 points
3. Commenting the code - 2 points
4. Explaining the mechanism that you use - 2 points
5. Adding fail-safety for the user input - 2 points

In this question, you can use the online compiler here.

Skriv in ditt svar här

```

//library with standard input output functions
#include <stdio.h>
//library with standard functions
#include <stdlib.h>
//library with standard time functions
#include <time.h>
//library with support for booleans
#include <stdbool.h>
//library with support for types
#include <ctype.h>
// library with standard string functions
#include <string.h>

// explicitly choosing variant 2 sort(if not defined, sort variant 1 gets chosen)
#define WHICHSORT TRUE

// if not defined, the sort executes below
#ifndef WHICHSORT
// YOU CANNOT MODIFY THE CODE BELOW
void sort(int arr[], int n) {
    int i, j, temp;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Swap arr[j] and arr[j+1]
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
// YOU ARE NOT ALLOWED TO MODIFY THE CODE ABOVE

//if it was defined, then sort variant 2 gets executed instead
#else
// YOU ARE NOT ALLOWED TO MODIFY THE CODE BELOW
void sort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        /* Move elements of arr[0..i-1], that are
           greater than key, to one position ahead
           of their current position */
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
// YOU ARE NOT ALLOWED TO MODIFY THE CODE ABOVE
//end of the if define clause, which chooses which sort variant to use
#endif

  /*
     MECHANISM:
     by using directives (i.e. preprocessor directives), I can decide which part of
         account, through
     the use of a flag. If the flag is defined, one sort variant will be included,

   */

// main program executing
int
main ()
 {

// seeding rand with current time
    srand ((unsigned int) time (NULL));

    //asking a number to the user
    printf ("Input number please.\n");

    // variable which will hold the size of the array
  int size;

    // scan user input for the size of the array, provide pointer in parameter
    scanf ("%d", &size);

    // checking if user provided a wrong input, such as negative number, 0, or text
    if (size < 1)
    {

printf
        ("Please make sure to provide a number (not text) that is positive, which w
            );

        // program ended with error
        return 1;

}

    //declaration of the array
  int bestArray[size];

printf ("Printing unsorted array\n");

    // loop through the array (empty so far)
    for (int i = 0; i < size; i++)

    {

        // create random numbers for the array
      int randomNumber = (int) rand ();

        //attach the number to the next position in the array
        bestArray[i] = randomNumber;
        // print the current number of the unsorted array
        printf ("%d ", bestArray[i]);

}
// sorting the array
    sort (bestArray, size);

printf ("\nPrinting sorted array\n");

    // iterating again through the array, but sorted this time
    for (int i = 0; i < size; i++)

    {

printf ("%d ", bestArray[i]);

}
// informing end of program
    printf ("\nEnd of program.");
    // program ended correctly.
    return 0;
}

```

Besvarad.