STUDENT

DIT63X-0020-OLU

TENTAMEN

DIT632-DIT633 june re-exam 2025 - L

Kurskod	--
Bedömningsform	--
Starttid	12.06.2025 12:00
Sluttid	12.06.2025 18:00
Bedömningsfrist	--
PDF skapad	26.11.2025 15:21
Skapad av	Lisa Lindén

Instructions for the exam

DIT 633 - Development of Embedded and Real-time systems

Welcome to this edtion of the DIT633 examination!

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. Please note that the log-in to TinkerCad is different this time, so please read the instruction carefully (available in the TinkerCad question), especially regarding your **class nickname and class code:**

Please use the following procedure to log in to TinkerCad:
- go to log-in
- choose "Students with Class Code"
*- use the class code "*IBM V6R XJU*"*
- use join with nickname. Your anonymous code is your nickname - without the dashes and using small letters, e.g., DIT633-0001-ABC becomes dit6330001abc as your nickname

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 Reading pointers

int (x)[]**

- x is pointer to pointer to array of function that returns an int
- x is a function that returns a pointer to pointer to array of int
- x is pointer to pointer to array of int ✓
- x is an array of pointer to pointer to int

int (x[])()**

- x is an array of pointer to pointer to int
- x is an array of pointer to pointer to function returning int ✓
- x is an array of function returning pointer to int
- x is a function returning array of pointer to pointer to int

int (*(*(*x)(void)))

- x is a pointer to function (void) returning pointer to pointer to int ✓
- x is a function (void) returning pointer to pointer to a pointer to int
- x is pointer to a pointer to a function (void) returning pointer to pointer to int
- x is a function (void) returning pointer to pointer to int ✗

int x(int *())

- x is a function (function returning array of pointer to int) returning int
- x is a function that takes as an argument a function returning pointer to int, and retu nt ✓
- x is a function (function returning pointer to int) returning pointer to int
- x is a function (function returning int) returning pointer to int

Delvis rätt. 0 av 0 poäng.

2 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.

In the implementation, you MUST use two different threads (one for Bunny and one for Jack), you must also coordinate/synchronize them.

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: 2 points
2. Using bit operations: 3 points
3. Commenting the code: 1 point
4. Using multithreading: 4 points

In this question, you can use the online compiler here

Skriv in ditt svar här

```c
/*******************************************************************************

Welcome to GDB Online.
  GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Rub
  C#, OCaml, VB, Perl, Swift, Prolog, Javascript, Pascal, COBOL, HTML, CSS, JS
  Code, Compile, Run and Debug online from anywhere in world.

*******************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>
#include <time.h>
#include <pthread.h>

#define ROUNDS 10

// shared state
static int  bunnyPos  = 0;
static int  jackPos   = 0;
static bool bunnyTurn = true;                    // whose turn to draw
pthread_mutex_t mtx = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t  cv  = PTHREAD_COND_INITIALIZER;

// extract `len` bits starting at bit `start` (0 = LSB)
static int extractBits(uint8_t byte, int start, int len) {
    return (byte >> start) & ((1u << len) - 1);
}

// thread function: each thread draws ROUNDS bytes, alternating turns
static void* drawThread(void* arg) {
    bool isBunnyThread = *(bool*)arg;
    const char* drawer  = isBunnyThread ? "Bunny" : "Jack";

    for (int round = 1; round <= ROUNDS; round++) {
        // wait for our turn
        pthread_mutex_lock(&mtx);
        while (bunnyTurn != isBunnyThread)
            pthread_cond_wait(&cv, &mtx);

        // draw a random byte
        uint8_t b = (uint8_t)(rand() & 0xFF);
        printf("Round %2d: %s draws: 0x%02X\n", round, drawer, b);

```

```
45          // decode:
46          // bit7 = who moves (1=Bunny,0=Jack)
47          // bit6 = direction (1=forward,0=backward)
48          // bits5..4 = multiplier 0..3 → +1
49          // bits3..0 = base steps 0..15
50          bool mover   = extractBits(b, 7, 1);
51          bool forward = extractBits(b, 6, 1);
52          int  mult    = extractBits(b, 4, 2) + 1;
53          int  steps   = extractBits(b, 0, 4) * mult;
54          if (!forward) steps = -steps;
55
56          // apply move
57          if (mover) bunnyPos += steps;
58          else       jackPos  += steps;
59
60          // print result
61          printf("     %s moves %d %s;  Bunny =%d, Jack=%d\n\n",
62                 mover ? "Bunny" : "Jack",
63                 abs(steps),
64                 forward ? "forward" : "backward",
65                 bunnyPos, jackPos);
66
67          // switch turn
68          bunnyTurn = !bunnyTurn;
69          pthread_cond_signal(&cv);
70          pthread_mutex_unlock(&mtx);
71      }
72      return NULL;
73  }
74
75  int main() {
76      srand((time(NULL)));
77      printf("Welcome to the Easter Bunny vs Jack Frost game!\n\n");
78
79      // initialise threads
80      bool tBunny = true, tJack = false;
81      pthread_t tidBunny, tidJack;
82      pthread_create(&tidBunny, NULL, drawThread, &tBunny);
83      pthread_create(&tidJack, NULL, drawThread, &tJack);
84
85      // wait for both to finish
86      pthread_join(tidBunny, NULL);
87      pthread_join(tidJack, NULL);
88
89      // final outcome
90      printf("Game Over!\n");
91      if       (bunnyPos > jackPos) printf("Easter Bunny wins!  (Bunny Position = %d
            jackPos);
92      else if (jackPos  > bunnyPos) printf("Jack Frost wins!    (Bunny Position = %
            jackPos);
93      else                          printf("It's a tie!         (Bunny Position = %
            jackPos);
94
95      return 0;
96  }
```

Besvarad.

3 Beach Volleyball Simulation - Paris 2024

In the Paris 2024 Olympics, the Beach Volleyball event introduces an innovative twist where the outcome of each point is determined by a computer simulation. The simulation uses randomly generated bytes to decide the actions of each team during a rally.

Each byte generated by the simulation represents a set of actions taken by a player from one of the teams. The byte is interpreted according to the following rules:

- **Bit 0 (MSB)**: Determines which team gets to act.
  - `1` - Team A.
  - `0` - Team B.
- **Bit 1**: Determines the type of action.
  - `1` - Spike (offensive).
  - `0` - Block (defensive).
- **Bits 2-3**: Define the power level of the action (multiplier).
- **Bits 4-7**: Define the effectiveness of the action (points scored or prevented).

For example, if the byte is `0xC9`, it would be interpreted as:

- Bit 0: `1` (Team A acts)
- Bit 1: `0` (Block action)
- Bits 2-3: `10` (Multiplier = 3)
- Bits 4-7: `1001` (Effectiveness = 9)

Thus, Team A performs a block with a power level of 3, preventing `3 * 9 = 27` points from being scored by the opposing team.

Your task:
Write a C program that simulates a Beach Volleyball game between Team A and Team B. The program should:

1. Randomly generate 10 bytes for each team.
2. Interpret each byte according to the rules above.
3. Track the total points scored by each team after each rally.
4. Print the results of each rally, showing which team acted, what type of action was taken, and how many points were scored or prevented.
5. Declare the winner based on the total points after 10 rallies.

Points: 10

- Random byte generation: 2 points
- Correct interpretation of the byte: 4 points
- Tracking and updating the score: 1 points
- Printing the rally-by-rally status: 2 points
- Commenting the winner correctly: 1 point

You are allowed to use the compiler from the resources below.

Skriv in ditt svar här

```
1  /*************************************************************************
2
```

```
Welcome to GDB Online.
  GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Rub
  C#, OCaml, VB, Perl, Swift, Prolog, Javascript, Pascal, COBOL, HTML, CSS, JS
  Code, Compile, Run and Debug online from anywhere in world.

*******************************************************************************/

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <time.h>

#define RALLIES 10

// Function to extract bits from a byte
static int extractBits(uint8_t byte, int start, int len) {
    return (byte >> start) & ((1u << len) - 1);
}

int main() {
    srand(time(NULL));  // Seed the random number generator

    int scoreA = 0;
    int scoreB = 0;

    printf("=== Beach Volleyball Simulation (Paris 2024) ===\n\n");

    for (int i = 0; i < RALLIES; i++) {
        uint8_t actionByte = rand() % 256;  // Random byte between 0 and 255

        int teamBit = extractBits(actionByte, 7, 1);         // Bit 0 (MSB)
        int actionBit = extractBits(actionByte, 6, 1);       // Bit 1
        int multiplier = extractBits(actionByte, 4, 2) + 1;  // Bits 2-3 (2 bits)
        int effectiveness = extractBits(actionByte, 0, 4);   // Bits 4-7

        int points = multiplier * effectiveness;

        // Print rally summary
        printf("Rally %2d: Byte = 0x%02X ", i + 1, actionByte);
        if (teamBit == 1) {
            // Team A acts
            if (actionBit == 1) {
                // Spike
                printf("Team A spikes (Power %d, Effectiveness %d) -> +%d points\
                    );
                scoreA += points;
            } else {
                // Block
                printf("Team A blocks (Power %d, Effectiveness %d) -> Team B -%d
                    points);
                scoreB -= points;
                if (scoreB < 0) scoreB = 0;
            }
        } else {
            // Team B acts
            if (actionBit == 1) {
                // Spike
                printf("Team B spikes (Power %d, Effectiveness %d) -> +%d points\
                    );
                scoreB += points;
            } else {
                // Block
                printf("Team B blocks (Power %d, Effectiveness %d) -> Team A -%d
                    points);
                scoreA -= points;
                if (scoreA < 0) scoreA = 0;
            }
        }
    }

    // Final scores
    printf("\n=== Final Scores ===\n");
    printf("Team A: %d points\n", scoreA);
    printf("Team B: %d points\n", scoreB);
```

```
73
74      // Declare winner
75      if (scoreA > scoreB) {
76          printf("Winner: Team A \n");
77      } else if (scoreB > scoreA) {
78          printf("Winner: Team B \n");
79      } else {
80          printf("It's a tie!\n");
81      }
82
83      return 0;
84  }
```

Besvarad.

4 Dynamic strings from command line

Write a program in C that reads 3-10 strings from the command line arguments, stores them in an array, and finds the shortest string in the array. The program should then remove the shortest string and return the array, now shorter by one element.

The number of strings should be dynamic, depending on the number of arguments provided.

If two strings share the same length, only the first one should be removed.

Requirements:

1. **Reading Strings:** The program should read 3-10 strings from the command line arguments and store them in an array. The number of strings that is read should be dynamic.
2. **Finding the Shortest String:** The program should identify the shortest string in the array.
3. **Removing the Shortest String:** Once the shortest string is identified, the program should remove it from the array.
4. **Returning the Modified Array:** After removing the shortest string, the program should return the array, now containing one less element.
5. **Displaying Output:** The program should write the string that has been removed to the console and display all elements of the array both before and after the removal of the shortest string.

Typical usage: **main.exe string1 string23 string_is_ok another_string loremIpsum new_string**

The program should contain a separate function to find and remove the shortest string. The program should have a main function that tests the functionality.

Grading Criteria:

- Correct functionality (as specified above): 3 points
- Comments: 2 points
- Function to find and remove the element: 2 points
- Main function to test the program: 2 points
- Safety checks (e.g., handling edge cases): 1 point

Skriv in ditt svar här

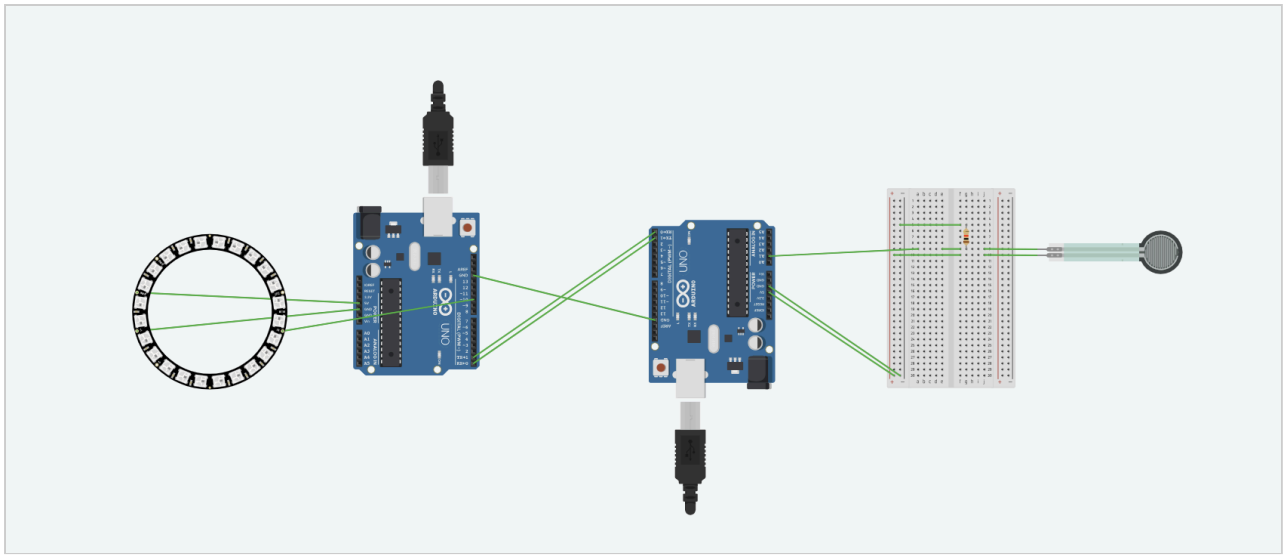
Skriv in ditt svar här

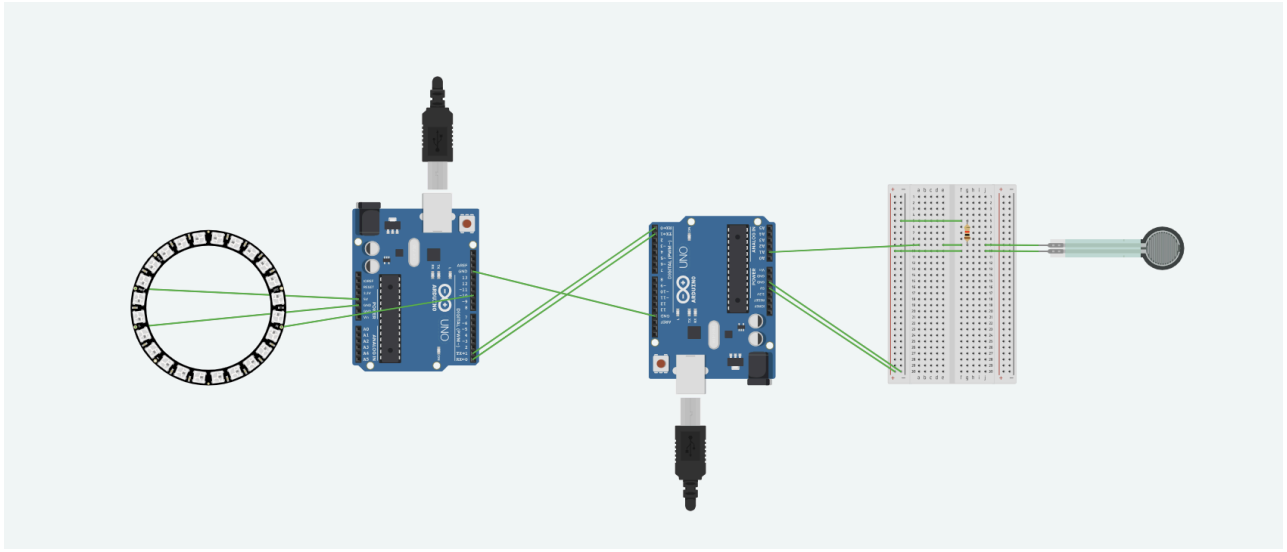
```
1  /*******************************************************************************
2
3  Welcome to GDB Online.
4    GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Rub
5    C#, OCaml, VB, Perl, Swift, Prolog, Javascript, Pascal, COBOL, HTML, CSS, JS
6    Code, Compile, Run and Debug online from anywhere in world.
7
8  *******************************************************************************/
9
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <string.h>
13
14 /**
15  * Function to find and remove the shortest string from an array.
```

```c
 * Function to find and remove the shortest string from an array.
 * If multiple strings have the same shortest length, only the first is removed.
 */
void findAndRemoveShortestString(char *arr[], int *size) {
    if (*size <= 0) {
        printf("No elements to remove.\n");
        return;
    }

    int minIndex = 0;
    int minLength = strlen(arr[0]);

    // Find index of the shortest string
    for (int i = 1; i < *size; i++) {
        int len = strlen(arr[i]);
        if (len < minLength) {
            minLength = len;
            minIndex = i;
        }
    }

    printf("\nRemoved string: %s\n", arr[minIndex]);

    // Shift remaining elements to the left
    for (int i = minIndex; i < *size - 1; i++) {
        arr[i] = arr[i + 1];
    }

    (*size)--; // Decrease size after removal
}

int main(int argc, char *argv[]) {
    // Ensure valid number of arguments: 3 to 10 strings
    if (argc < 4 || argc > 11) {
        printf("Error: Please provide between 3 and 10 strings as command line ar
        return 1;
    }

    // `argv[1]` to `argv[argc - 1]` are the user strings
    int size = argc - 1;
    char **strings = &argv[1];

    // Display original strings
    printf("Strings before removal:\n");
    for (int i = 0; i < size; i++) {
        printf("%s\n", strings[i]);
    }

    // Remove the shortest string
    findAndRemoveShortestString(strings, &size);

    // Display remaining strings
    printf("\nStrings after removal:\n");
    for (int i = 0; i < size; i++) {
        printf("%s\n", strings[i]);
    }

    return 0;
}
```

Besvarad.

5 Two arduinos and force sensor





In this circuit you need the implement a system where one Arduino (right-hand side) is responsible for sensing, i.e., information from the force sensor, and the other Arduino (left-hand side) is responsible for actuating (neopixel ring).

The system should work as follows:

1. When the force is applied to the force sensor, the sensing Arduino reads the value and send the serial message to the actuating Arduino.
2. The message send between Arduinos should be max 1 byte.
3. When no force is applied to the sensor, the message should contain 0
4. When the actuating Arduino receives a message, it should turn on the number of pixels that correspond to the force applied - scaled to the number of pixels available.
5. When the maximum force is applied, the neopixel ring should blink with all LEDs until the user releases the force sensor.

In this question you are allowed to use TinkerCad, in the resource link below.

Please use the following procedure to log in to TinkerCad:

- go to log-in

- choose "Students with Class Code"

*- use the class code "*PHG KPL MMZ*"*

- use join with nickname. Your anonymous code is your nickname - without the dashes and using small letters, e.g., DIT63x-0001-ABC becomes dit63x0001abc as your nickname

Grading:

- implementing the sensing logic: 2 points
- implementing the actuation logic: 2 points
- implementing the serial communication: 2 points
- commenting the code: 2 points

Skriv in ditt svar här

```
#include <Adafruit_NeoPixel.h>

// C++ code
//
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Besvarad.