קבצי המקור של נמל **חיפה**:

שם: protocol.h

תפקיד: common header – (זהה לאילת וחיפה) מגדיר קבועים המשותפים לקוד תהליך האב והבן (קודים של שגיאות, קבועים של פרוטוקול הIPC לתקשורת בין התהליכים) ומימוש של פונקציות שנעשה בהן שימוש רב בקוד (קריאת זמן, הדפסות, דיווח על שגיאות)

code:

///////// error codes //////////////////

// Could have use macros from <winerror.h> for built in types.

#define VESSEL\_ID\_ERROR 2

#define NUM\_VESSELS\_ERROR 3

#define MEMORY\_ALLOCATION\_ERROR 4

#define PIPE\_CREATION\_ERROR 5

#define PROCESS\_CREATION\_ERROR 6

#define PIPE\_USAGE\_ERROR 7

#define THREAD\_CREATION\_ERROR 8

#define UNKNOWN\_ANSWER\_CODE\_ERROR 8

#define MUTEX\_RELEASE\_ERROR 9

#define MUTEX\_CREATION\_ERROR 10

#define SEMAPHORE\_CREATION\_ERROR 11

#define SEMAPHORE\_RELEASE\_ERROR 12

#define PIPE\_HANDLE\_ERROR 13

#define INVALID\_CRANE\_IDX\_ERROR 14

#define OPEN\_MUTEX\_ERROR 15

//////////////////////////////////////////

// CONSTANTS ////////////////////////////

#define MIN\_VESSELS 2

#define MAX\_VESSELS 50

#define PIPE\_BUF\_LEN 50

#define OUTPUT\_BUFFER\_LEN 256

//random time between 5 and 3000 msec between actions

#define MIN\_SLEEP\_TIME 5

#define MAX\_SLEEP\_TIME 3000

#define BUFSIZE 4096 //4KB- Up till this size, we are guaranteed that writes to pipe will be atomic

#define INVALID\_VESSEL\_ID (-1)

// pipe communication codes

#define VESSELS\_NOT\_ALLOWED "N"

#define VESSELS\_ALLOWED "Y"

#define DONE\_INDICATOR '#'

#define PRINT\_INDICATOR '$'

#define PRINT\_MUTEX "printingMutex"

////////////////////////////////////////

SYSTEMTIME timeToPrint;

//generic function to print system time information

const char\* getTimeString()

{

GetLocalTime(&timeToPrint);

static char currentLocalTime[20];

sprintf(currentLocalTime, "%02d:%02d:%02d", timeToPrint.wHour, timeToPrint.wMinute, timeToPrint.wSecond);

return currentLocalTime;

}

void printToLog(char\* msg) {

// Note that we use stderr here in order to print to the shared parent console

// printing to stdout won't work (see the redirections at initializePipesAndSI() )

fprintf(stderr, "[%s] %s\n", getTimeString(), msg);

return;

}

//generic function to randomise a Sleep time between MIN\_SLEEP\_TIME and MAX\_SLEEP\_TIME msec

int randSleepTime() {

int sleepTime = rand() % (MAX\_SLEEP\_TIME - MIN\_SLEEP\_TIME) + MIN\_SLEEP\_TIME;

return sleepTime;

}

שם: **HaifaPort.c**

תפקיד: מימוש הלוגיקה של תהליך האב:

- אחראי על קבלת ארגומנט מספר הספינות משא מהמשתמש ובדיקה ראשונית שלו (האם הוא בין 2 ל 50)

- יצירת 2 צינורות אנונימיים לתקשורת חד כיוונית

- הגדרה של מה יעבור בירושה לתהליך הבן והפניה מחדש(redirection) של ערוץ הקלט והפלט הסטנדרטיים (file descriptors) של תהליך הבן שייווצר עם הhandles לגישה לצינורות מהצד של אילת, ושל ערוץ השגיאות הסטנדרטי של הבן עם זה של האב (הרי לא ניתן לשתף את הhandles של הצינורות על ידי שליחה על גבי צינור משום שתהליך הבן לא ידע מאיפה לקרוא –"הביצה והתרנגולת" משום כך יש צורך בטריק כלשהו של שיתוף מידע בעת יצירת תהליך הבן)

לקריאה נוספת באתר של מייקרוסופט :   
<https://docs.microsoft.com/en-us/windows/win32/procthread/creating-a-child-process-with-redirected-input-and-output>

- יצירת תהליך הבן (עבור נמל אילת).

- העברת ארגומנט מספר הספינות על גבי צינור שליחה.

- יצירת תהליכונים עבור כל הספינות ושליחתן לאילת (בצורה של העברת הודעה על צינור השליחה).

- המתנה להודעות על גבי צינור הקבלה שיסמלו את חזרת הספינות לחיפה. ויציאה מהתוכנית לאחר מכן

- הדפסת הודעות למסך לצרכי logging.

Code:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <time.h>

#include <windows.h>

#include "protocol.h"

// TODO: Can't actually use it (Works only with CreateProcessA())

#define EILAT\_PROCESS "EilatPort.exe"

char buffer[PIPE\_BUF\_LEN];

DWORD WINAPI Vessel(PVOID);

// Pipes handles (2x2)

static HANDLE HaifaToEilatReadH, HaifaToEilatWriteH, EilatToHaifaReadH, EilatToHaifaWriteH;

static HANDLE haifaMutex; //(\*)Currently the pipe mutex is used only in this function.

// but we might want to use it in the future in other functions

static HANDLE printMutex; //mutual mutex for Eilat and Haifa for printing to the console

static int\* vesselsIDsArr; //see (\*)

static HANDLE\* vesselsSemaphoreArr; // For synchronization between vessel in Haifa & Eilat

// a vessel thread must finish only when it got back from Eilat

HANDLE\* vesselsThreads;

//Functions declarations

void initStaticVars();

void Sails\_to\_Eilat(int);

void initializePipesAndSI();

void createVesselsThreads();

void waitForAllVesselsToReturn();

void joinAllVessels();

void closeResources();

void consolePrint(char\*);

void exitWithError(const char\* msg, int errorCode);

static int numOfVessels = 0;

static STARTUPINFO si;

static PROCESS\_INFORMATION pi;

int main(int argc, char\* argv[])

{

if (argc != 2)

{

printf("Usage: HaifaPort.exe numOfVessels");

exit(1);

}

numOfVessels = atoi(argv[1]); // TODO: use strtol instead

if ((numOfVessels < MIN\_VESSELS) || (numOfVessels > MAX\_VESSELS))

{

printf("First arg (numOfVessels) must be between %u and %u", MIN\_VESSELS, MAX\_VESSELS);

exit(0);

}

printf("[%s] HaifaPort: Number of vessels is %d\n", getTimeString(), numOfVessels);

srand((unsigned int)time(NULL));

// This must be done before creating the child process!

// because we want it to share the pipes handles with it

initializePipesAndSI(); //Initialize pipes & handles

/\* Creating the child process - EilatPort \*/

//TODO: There might be a better way to do this with a EXE\_PATH macro.

// But it works with CreateProcess()

TCHAR ProcessName[256]; //TCHAR is either char or WCHAR based on wether UNICODE is defined

wcscpy(ProcessName, L"EilatPort.exe"); // prepare the process name argument

// L is for wchar\_t literal string

//CHAR str[] = EILAT\_PROCESS;

//LPSTR param = str;

// Just in case the generated exe name won't match the intended one- we will do 2 more trials.

int numberOfTries = 0;

while (numberOfTries < 3) {

//TODO: FILE\_PATH MACRO works only with CreateProcessA() which is less recommended

if (!CreateProcess(NULL, // No module name (use command line).

ProcessName,// Command line.

NULL, // Process handle not inheritable.

NULL, // Thread handle not inheritable.

TRUE, // Set handle inheritance to true (a must here)

0, // No creation flags. By default, a console process inherits its parent's console.

NULL, // Use parent's environment block.

NULL, // Use parent's starting directory.

&si, // Pointer to STARTUPINFO structure.

&pi) // Pointer to PROCESS\_INFORMATION structure.

)

{

numberOfTries++;

numberOfTries == 1 ? wcscpy(ProcessName, L"Eilat.exe") : wcscpy(ProcessName, L"EilatSrc.exe");

}

else {

break; //sucess

}

}

if (numberOfTries == 3)

exitWithError("Process Creation Failed", PROCESS\_CREATION\_ERROR);

consolePrint("HaifaPort: EilatPort process has been created");

consolePrint("HaifaPort: Requesting permission from EilatPort to pass vessels");

DWORD bytesSent, bytesRecieved;

/\* Send EilatPort the NumOfVessels in HaifaPort through the relevant anonymous pipe \*/

// Note: We send the value as a string

if (!WriteFile(HaifaToEilatWriteH, argv[1], PIPE\_BUF\_LEN, &bytesSent, NULL))

exitWithError("HaifaPort: Failed to send value of number of vessels to Eilat", PIPE\_USAGE\_ERROR);

/\* Wait for response from EilatPort \*/

if (!ReadFile(EilatToHaifaReadH, buffer, PIPE\_BUF\_LEN, &bytesRecieved, NULL))

exitWithError("HaifaPort: Error reading from pipe while waiting for confirmation from EilatPort", PIPE\_USAGE\_ERROR);

if (strcmp(buffer, VESSELS\_NOT\_ALLOWED) == 0)

{

exitWithError("HaifaPort: request wasn't approved, number of vessels invalid!", NUM\_VESSELS\_ERROR);

}

else if (strcmp(buffer, VESSELS\_ALLOWED) == 0)

{

consolePrint("HaifaPort: request to pass was approved by EilatPort");

initStaticVars();

createVesselsThreads();

waitForAllVesselsToReturn();

}

else

{

exitWithError("HaifaPort: received an unknown answer code from EilatPort", UNKNOWN\_ANSWER\_CODE\_ERROR);

}

// Wait until child process (EilatPort) exits.

WaitForSingleObject(pi.hProcess, INFINITE);

joinAllVessels(); //wait till all Haifa's Vessel threads are done

consolePrint("Haifa Port: Haifa Port: Exiting...");

closeResources();

exit(0); //TODO: could use return instead

}

void exitWithError(const char\* msg, int errorCode) {

WaitForSingleObject(printMutex, INFINITE);

fprintf(stderr, "[%s] Error - %s (%d)\n", getTimeString(), msg, GetLastError());

if (!ReleaseMutex(printMutex)) {

fprintf(stderr, "Error: Failed to release printing mutex\n %d \n", GetLastError());

exit(MUTEX\_RELEASE\_ERROR);

}

exit(errorCode);

}

//Thread function for each Vessel

DWORD WINAPI Vessel(PVOID Param)

{

char convertToPrint[OUTPUT\_BUFFER\_LEN];

//Get the Unique ID from Param and keep it in a local variable

int VesselID = \*(int\*)Param;

sprintf(convertToPrint, "Vessel %d - starts sailing @ Haifa Port", VesselID);

consolePrint(convertToPrint);

Sleep(randSleepTime());

Sails\_to\_Eilat(VesselID);

// Wait till the vesssel gets back to HaifaPort

WaitForSingleObject(vesselsSemaphoreArr[VesselID - 1], INFINITE);

sprintf(convertToPrint, "Vessel %d - done sailing @ Haifa Port", VesselID);

consolePrint(convertToPrint);

return 0;

}

//function that send the vessels frome Haifa

void Sails\_to\_Eilat(int Vesselid) {

char convertToPrint[OUTPUT\_BUFFER\_LEN];

char message[PIPE\_BUF\_LEN];

DWORD bytesSent;

//Access to state DB is done exclusively (protected by mutex)

WaitForSingleObject(haifaMutex, INFINITE);

sprintf(message, "%d", Vesselid);

sprintf(convertToPrint, "Vessel %d - entering Canal: Med. Sea ==> Red Sea", Vesselid);

consolePrint(convertToPrint);

Sleep(randSleepTime());

/\* the parent now wants to write to the pipe \*/

if (!WriteFile(HaifaToEilatWriteH, message, PIPE\_BUF\_LEN, &bytesSent, NULL))

fprintf(stderr, "Error writing to pipe \n %d \n", GetLastError());

if (!ReleaseMutex(haifaMutex))

fprintf(stderr, "HaifaPort -> CrossToEilat::Unexpected error mutex.V(), error num - %d\n", GetLastError());

}

//Write protection function of two Process.

void consolePrint(char\* msg)

{

WaitForSingleObject(printMutex, INFINITE);

printToLog(msg);

if (!ReleaseMutex(printMutex)) {

fprintf(stderr, "HaifaPort: Error - Failed to release printing mutex\n %d \n", GetLastError());

exit(MUTEX\_RELEASE\_ERROR);

}

}

//Function for allocating and initializing memory for static variables

void initStaticVars() {

haifaMutex = CreateMutex(NULL, FALSE, NULL);

if (!haifaMutex)

{

exitWithError("Error CreateMutex() failed Haifa's Mutex", MUTEX\_CREATION\_ERROR);

}

vesselsIDsArr = (int\*)malloc(numOfVessels \* sizeof(int));

if (vesselsIDsArr == NULL)

{

exitWithError("Error: malloc for Vessels IDs array failed", MEMORY\_ALLOCATION\_ERROR);

}

// see https://stackoverflow.com/questions/8224347/malloc-vs-heapalloc

// in this project we use HeapAlloc for Win32API types

// and malloc for standart c types

// Dynamically allocated and clear(zero) memory for vessels semaphores array

vesselsSemaphoreArr = (HANDLE\*)HeapAlloc(GetProcessHeap(), HEAP\_ZERO\_MEMORY, (numOfVessels \* sizeof(HANDLE)));

if (!vesselsSemaphoreArr)

{

exitWithError("Error: malloc for Vessels semaphores array failed", MEMORY\_ALLOCATION\_ERROR);

}

vesselsThreads = (HANDLE\*)HeapAlloc(GetProcessHeap(), HEAP\_ZERO\_MEMORY, (numOfVessels \* sizeof(HANDLE)));

if (!vesselsThreads)

{

exitWithError("Error: malloc for Vessels array failed", MEMORY\_ALLOCATION\_ERROR);

}

for (int i = 0; i < numOfVessels; i++)

{

vesselsSemaphoreArr[i] = CreateSemaphore(NULL, 0, 1, NULL);

if (!vesselsSemaphoreArr[i])

{

exitWithError("Error: CreateSemaphore() for a vessel failed", SEMAPHORE\_CREATION\_ERROR);

}

}

return;

}

//Initialization the pipes and handles

void initializePipesAndSI()

{

/\* set security attributes such that pipe handles are inherited (bInheritHandle = true)\*/

SECURITY\_ATTRIBUTES sa = { sizeof(SECURITY\_ATTRIBUTES), NULL,TRUE };

ZeroMemory(&pi, sizeof(pi));

// Create the printing mutex

printMutex = CreateMutex(NULL, FALSE, TEXT(PRINT\_MUTEX));

if (!printMutex)

{

fprintf(stderr, "Error: CreateMutex failed");

exit(MUTEX\_CREATION\_ERROR);

}

/\* Create an unnamed pipe for Haifa->Eilat communication \*/

if (!CreatePipe(&HaifaToEilatReadH, &HaifaToEilatWriteH, &sa, 0)) {

exitWithError("CreatePipe() for Haifa to Eilat failed", PIPE\_CREATION\_ERROR);

}

// Ensure the write handle to the pipe for StdinWrite is not inherited.

if (!SetHandleInformation(HaifaToEilatWriteH, HANDLE\_FLAG\_INHERIT, 0)) {

exitWithError("WriteHandleIn SetHandleInformation", PIPE\_HANDLE\_ERROR);

}

/\* Create an unnamed pipe for Eilat->Haifa communication\*/

if (!CreatePipe(&EilatToHaifaReadH, &EilatToHaifaWriteH, &sa, 0)) {

exitWithError("CreatePipe() for Eilat to Haifa failed", PIPE\_CREATION\_ERROR);

}

// Ensure the read handle to the pipe for STDOUT is not inherited.

if (!SetHandleInformation(EilatToHaifaReadH, HANDLE\_FLAG\_INHERIT, 0)) {

exitWithError("HaifaPort: Failed to SetHandleInformation for ReadHandleOut", PIPE\_HANDLE\_ERROR);

}

// Set up members of the STARTUPINFO structure.

/\* Establish the START\_INFO structure for the child process \*/

ZeroMemory(&si, sizeof(si));

GetStartupInfo(&si);

si.cb = sizeof(STARTUPINFO);

// This structure specifies the STDIN ,STDOUT and STDERR handles for redirection.

// Configure startup info for the child process so that

// we will have a redirection/overriding of the child process stdin & stdout

// with the relevant pipes handles (On Eilat side)

// This is a simple way to "send" it the handles at creation time

// But it assumes the child process stdin and stdout won't be used

// Because anonymouse pipe has special methods to read from and write to it

// (ReadFile, WriteFile), [

si.hStdInput = HaifaToEilatReadH;

si.hStdOutput = EilatToHaifaWriteH;

// we wish to share our console (stderr) with the child process

//We want to be able to show prints that are done in EilatPort

si.hStdError = GetStdHandle(STD\_ERROR\_HANDLE);

si.dwFlags = STARTF\_USESTDHANDLES;

}

// wait for all Vessel threads in Haifa to finish .

void joinAllVessels()

{

char convertToPrint[OUTPUT\_BUFFER\_LEN];

// Wait for all vessels threads to finish

WaitForMultipleObjects(numOfVessels, vesselsThreads, true, INFINITE); //bWaitAll =true, no timeout

sprintf(convertToPrint, "Haifa Port: All Vessel Threads are done");

consolePrint(convertToPrint);

}

void closeResources() {

//Close the pipes handles on Haifa's side

CloseHandle(HaifaToEilatWriteH);

CloseHandle(EilatToHaifaReadH);

CloseHandle(haifaMutex);

CloseHandle(printMutex);

for (int i = 0; i < numOfVessels; i++)

{

CloseHandle(vesselsSemaphoreArr[i]);

CloseHandle(vesselsThreads[i]);

}

free(vesselsIDsArr);

HeapFree(GetProcessHeap(), HEAP\_NO\_SERIALIZE, vesselsThreads);

HeapFree(GetProcessHeap(), HEAP\_NO\_SERIALIZE, vesselsSemaphoreArr);

CloseHandle(pi.hProcess);

CloseHandle(pi.hThread);

}

// Create Vessels threads.

void createVesselsThreads()

{

for (int i = 0; i < numOfVessels; i++)

{

vesselsIDsArr[i] = i + 1;

// Create Vessel threads and define their jobs as programmed in Vessel.

DWORD ThreadId; // In case we need the thread id (in this implementation we don't)

vesselsThreads[i] = CreateThread(NULL, 0, Vessel, &vesselsIDsArr[i], 0, &ThreadId);

if (!vesselsThreads[i])

{

char formattedStringBuff[OUTPUT\_BUFFER\_LEN];

sprintf(formattedStringBuff, "HaifaPort: Failed to create a thread for Vessel %d", i);

exitWithError(formattedStringBuff, THREAD\_CREATION\_ERROR);

}

}

}

// Read all Vessels Threads Return from Eilat. Report if Error occurred!

void waitForAllVesselsToReturn()

{

int completedVessels = 0;

char logBuffer[OUTPUT\_BUFFER\_LEN];

DWORD bytesRecieved;

int returnedVesselID = INVALID\_VESSEL\_ID;

// Serial waiting, no need to synchronize.

// We know exactly how many vessels are expected to come back

while (completedVessels < numOfVessels)

{

// using reading from pipe as a (blocking call) synchronization mechanism

if (!ReadFile(EilatToHaifaReadH, buffer, PIPE\_BUF\_LEN, &bytesRecieved, NULL))

{

exitWithError("HaifaPort: error reading from pipe", PIPE\_USAGE\_ERROR);

}

else if ((returnedVesselID = atoi(buffer)) != 0)

{

sprintf(logBuffer, "Vessel %s - exiting Canal: Red Sea ==> Med. Sea", buffer);

consolePrint(logBuffer);

// Wake up the waiting Haifa's Vessel thread (it arrived back in Haifa)

if (!ReleaseSemaphore(vesselsSemaphoreArr[returnedVesselID - 1], 1, NULL)) {

sprintf(logBuffer, "HaifaPort: ReleaseSemaphore error for vessel [%d]", returnedVesselID);

consolePrint(logBuffer);

}

completedVessels++;

}

else {

exitWithError("HaifaPort::waitForAllVesselsToReturn - atoi failed to interpert the returnedVesselID", VESSEL\_ID\_ERROR);

}

}

return;

}

קבצי המקור של נמל **אילת**:

**שם: protocol.h –**

העתק זהה אחד לאחד בתפקיד ובתוכן של הקובץ בעל שם זהה שמיובא על ידי נמל חיפה (HaifaPort.c) [הרעיון היה להכניס לשם רק את מה שמשותף לאילת וחיפה מבחינת הפרוטוקול] ולכן מיותר לצרף שוב את הקוד.

**שם: EilatPort.c**

**תפקיד**: מימוש הלוגיקה של תהליך נמל אילת

- קריאת ושמירת הhandles לצינורות שתהליך נמל חיפה יצר

- פתיחת הmutex המשותף לשני התהליכים לצורך סנכרון בין הדפסות למסך (יש כמה דרכים לשתף אותו)

- קריאת מספר הספינות מהצינור

-הגרלת מספר Cranes – עגורנים באופן כזה שיהיה מחלק של מספר הספינות

-יצירת תהליכי Vessels(ספינות) בנמל אילת שידמו את אלו שהגיעו מחיפה

-יצירת תהליכי Cranes(מנופים/עגורנים)

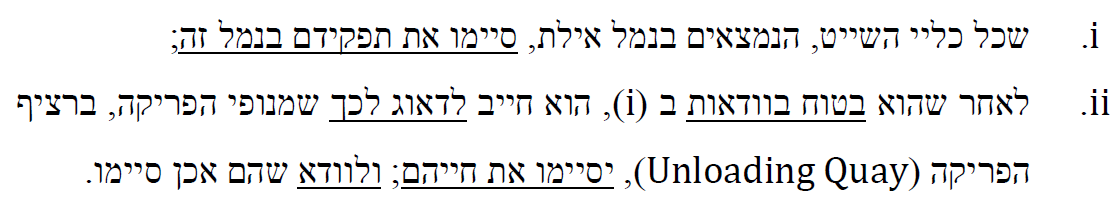
- סינון מעבר של הספינות אל רציף הפריקה בעזרת מחסום/נקודת מפגש (Barrier) המווסת כך שרק כאשר רציף הפריקה ריק ויש מספיק ספינות במחסום (כמספר הCranes) ישוחררו M=NumOfCranes ספינות לפריקה לפי First in First Out.

- הגדרת הלוגיקה של מטלת התהליכון (העבודה) עבור Vessel ו Crane

כל עגורן יודע מראש כמה ספינות עליו לשרת בסה"כ בצורה סדרתית. הוא ימתין בכל פעם עד להגעה של ספינה ולאחר מכן יתחיל לפרוק את המטען שלה.

* כל ספינה משוטטת בכוחות עצמה בנמל ואילו שאר הרכיבים (הנשלטים ע"י תהליך הנמל, ותהליכון העגורן) מונעים או מאפשרים את המעבר שלה בין התחנות בנמל. בסיום עבודתה היא חוזרת לנמל חיפה (נשלחת הודעה בצינור)

- ניקוי המשאבים וסיום התהליך לאחר שמתקיימים שני תנאי סיום.



(הקוד ממומש כך שלא ייתכן מצב שבו התהליכונים האלו לא סיימו את עבודתם בצורה תקינה ותהליך הבן של נמל אילת מסתיים "בהצלחה"/באופן תקין משום שתבוצע הקרסה של התהליך ודיווח שגיאה. לכן אפשר לומר בעירבון מוגבל שזה היה בסדר לא לבדוק את קוד החזרה של התהליכונים בעת המתנה לסיומם.)

קוד:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <math.h>

#include <time.h>

#include <windows.h>

#include "protocol.h"

#define CRANE\_AVAILABLE (-1)

#define INVALID\_CRANE\_IDX (-1)

#define MIN\_CARGO\_WEIGHT 5

#define MAX\_CARGO\_WEIGHT 50

static char readPipeBuffer[PIPE\_BUF\_LEN];

static HANDLE WriteHandle, ReadHandle;

static DWORD bytesSent, bytesRecieved;

static DWORD WINAPI Vessel(PVOID); //used for creating Vessel thread

static DWORD WINAPI Crane(PVOID); //used for creating Crane thread

static HANDLE\* Cranes;

static int num\_of\_cranes = 0;

static int cranesWithVessels = 0;

static int\* craneIDsArr;

static int\* cranesVessleIDs;

static int\* cargoWeights;

static int numOfVessels, vesselsInBarrier = 0;

static bool isQuayADTFree = true;

static int\* vesselThreadsIDArr;

static HANDLE\* vesselThreads;

static DWORD ThreadId;

static HANDLE sharedPrintMutex;

static HANDLE writePipeMutex, mutexAdt;

// TODO: Is there a better way to conrtol acception of a vessel to a crane and releasing it from it?

static HANDLE\* cranesOutSemaphores; //For synchronization between cranes and quay leaving vessels

static HANDLE\* cranesInSemaphores; //For synchronization between cranes and vessels entering the unloading quay

static HANDLE barrierMutex; //For defending the barrier data-structure

static HANDLE\* barrierQueue; // Array of semaphores for the barrier queue

// Possible space optimization for large numOfVessels: Use a linked list instead of an array

static int barrierTaillIdx = -1; //Last entered vessel array index of the barrier fifo

static int barrierHeadIdx = 0; // First in the barrier fifo to release next

//Functions prototypes

bool isPrime(int);

void checkNumOfVessels();

int generateNumberOfCranes(int);

void initStaticVars();

void logVesselAtBarrier(int);

void setPipesHandles();

void receiveVesselsFromHaifa();

void enterToBarrier(int);

void releaseFromBarrier();

int enterUnloadingQuay(int);

void leaveQuay(int, int);

void sailBackToHaifa(int);

void waitForCleanUpCondition();

void releaseResources();

void consolePrint(char\*);

void exitWithError(const char\* msg, int errorCode);

// Print error to mutual console , cleanup resources and exit

void exitWithError(const char\* msg, int errorCode) {

WaitForSingleObject(sharedPrintMutex, INFINITE);

fprintf(stderr, "[%s] Error - %s (%d)\n", getTimeString(), msg, GetLastError());

if (!ReleaseMutex(sharedPrintMutex)) {

fprintf(stderr, "Error: Failed to release printing mutex\n %d \n", GetLastError());

exit(MUTEX\_RELEASE\_ERROR);

}

releaseResources();

exit(errorCode); // ExitProcess accepts an unsigned int exit code (unlink exit(int))

}

//Thread function for each Crane

DWORD WINAPI Crane(PVOID Param)

{

//Get the Unique ID from Param and keep it in a local variable

int craneID = \*(int\*)Param;

char strBuffer[OUTPUT\_BUFFER\_LEN];

sprintf(strBuffer, "Crane - %d has been created", craneID);

consolePrint(strBuffer);

unsigned int vesselsPerCrane = numOfVessels / num\_of\_cranes;

//Every Crane will serve a total of vesselsPerCrane vessels one at a time

for (unsigned int i = 0; i < vesselsPerCrane; i++)

{

// Defends against Doing a Crane job until next time barrier releases vessels

WaitForSingleObject(cranesInSemaphores[craneID - 1], INFINITE);

int vesselID = cranesVessleIDs[craneID - 1];

sprintf(strBuffer, "vessel %d - Starts unloading %d Tons at Crane %d...", vesselID, cargoWeights[craneID - 1], craneID);

consolePrint(strBuffer);

Sleep(randSleepTime());

sprintf(strBuffer, "vessel %d - Finished unloading cargo of weight %d Tons at Crane %d", vesselID, cargoWeights[craneID - 1], craneID);

consolePrint(strBuffer);

//only now release vessel from crane

if (!ReleaseSemaphore(cranesOutSemaphores[craneID - 1], 1, NULL))

exitWithError("UnloadingQuay::Unexpected error cranesOutSemaphores.V()", SEMAPHORE\_RELEASE\_ERROR);

}

return 0; // Indication value for the related vessel that the unloading is done successfuly

}

//Thread function for each Vessel

DWORD WINAPI Vessel(PVOID Param)

{

//a unique Vessel ID equal to the one given in Haifa port

int vesselID = \*((int\*)Param);

// Before entering the Crane, The vessel need to wait at the barrier

enterToBarrier(vesselID);

Sleep(randSleepTime()); // simulate work

int craneIdx = enterUnloadingQuay(vesselID);

//Error Handling

if (craneIdx == INVALID\_CRANE\_IDX)

{

exitWithError("Unexpected Error Thread %d Entering Crain", INVALID\_CRANE\_IDX\_ERROR);

}

// need to wait for crane to finish it's job.

WaitForSingleObject(cranesOutSemaphores[craneIdx], INFINITE);

leaveQuay(vesselID, craneIdx); //release the vessel from the crane & quay

sailBackToHaifa(vesselID); //report sailing back and sail

return 0; //success ,finish the thread life

}

bool isPrime(int numOfVessels)

{

for (int i = 2; i <= (int)sqrt(numOfVessels); i++)

{

if (numOfVessels % i == 0)

{

return false;

}

}

return true;

}

// We need a random number of cranes such that:

// #vessels % #cranes == 0 and 1 < #cranes < #vessels

int generateNumberOfCranes(int numOfVessels) {

int numOfCranes;

while (true)

{

//possible matching values are in range [2 , numOfVessels/2]

numOfCranes = rand() % ((numOfVessels / 2) - 1) + 2;

if (numOfVessels % numOfCranes == 0)

{

return numOfCranes;

}

}

}

//Vessel has finished unloading in Eilat,and returns to back to Haifa.

void sailBackToHaifa(int vesselID)

{

char strBuffer[OUTPUT\_BUFFER\_LEN];

sprintf(strBuffer, "Vessel %d - entering Canal: Red Sea ==> Med. Sea ", vesselID);

consolePrint(strBuffer);

char tempWriteBuff[PIPE\_BUF\_LEN];

sprintf(tempWriteBuff, "%d", vesselID);

Sleep(randSleepTime()); // sleep to simulate sailing to Haifa

// send the vessel through the cannal to Haifa

WaitForSingleObject(writePipeMutex, INFINITE);

/\* Critical section start \*/

if (!WriteFile(WriteHandle, tempWriteBuff, PIPE\_BUF\_LEN, &bytesSent, NULL))

exitWithError("Eilat ::Error writing to pipe", PIPE\_USAGE\_ERROR);

/\* Critical section end \*/

if (!ReleaseMutex(writePipeMutex))

exitWithError("Eilat :: Unexpected error mutexPipe.V()", MUTEX\_RELEASE\_ERROR);

}

// A traffic passage flow control point for the vessels

// The Barrier is implemented as a queue (represented by a dynamic array)

// of binary semaphores [max queue size is equal to its capacity which is numOfVessels]

void enterToBarrier(int vessel\_ID)

{

char strBuffer[OUTPUT\_BUFFER\_LEN];

/\* This is a critical point! (deadlock danger)

Even though logically we might think we can postpone the lock of

mutexAdt till dealing with isQuayADTFree,

If we dont lock in the following order (SAME as the one in other functions)

We will have a deadlock =>Conclusion: Grab(Lock) multiple locks in the same order

We need two locks because the barrier and the ADT are dependent on each other's state

for some of their actions \*/

WaitForSingleObject(mutexAdt, INFINITE);

WaitForSingleObject(barrierMutex, INFINITE);

vesselsInBarrier++;

int barrierTaillIdxSnapshot = ++barrierTaillIdx; // the static var value might be changed after releasing barrier mutex

logVesselAtBarrier(vessel\_ID);

// Now there is e chance that there are enough vessels in barrier:

// If the unloading quay is empty AND There are at least numOfCranes vessels in the barrier:

// then release the group from the barrier

if (isQuayADTFree && vesselsInBarrier >= num\_of\_cranes) {

releaseFromBarrier(); // we have enough vessels at the barrier and unloading Quay is free

}

/\* Order of unlocking is less important here and won't create a deadlock

But we prefer releasing in the reverse order \*/

if (!ReleaseMutex(barrierMutex))

exitWithError("EilatPort :: Unexpected error while trying to release barrier mutex", MUTEX\_RELEASE\_ERROR);

if (!ReleaseMutex(mutexAdt))

exitWithError("EilatPort :: Unexpected error while trying to release mutexAdt", MUTEX\_RELEASE\_ERROR);

// Wait to enter the Unloading quay

WaitForSingleObject(barrierQueue[barrierTaillIdxSnapshot], INFINITE);

/// Enter critical section

WaitForSingleObject(barrierMutex, INFINITE);

vesselsInBarrier--;

if (!ReleaseMutex(barrierMutex))

exitWithError("EilatPort :: Unexpected error while trying to release barrier mutex", MUTEX\_RELEASE\_ERROR);

/// Exit critical section

sprintf(strBuffer, "Vessel %d - Released from barrier to unloading quay", vessel\_ID);

consolePrint(strBuffer);

}

//Function for process of Vessel with vesselID entry to the unloadingQuay.

//it settles down at an available Crane for unloading and wakes up the relevant crane.

int enterUnloadingQuay(int vessel\_ID)

{

char strBuffer[OUTPUT\_BUFFER\_LEN];

int craneIdx = INVALID\_CRANE\_IDX;

/\* Enter critical section \*/

WaitForSingleObject(mutexAdt, INFINITE);

//Find an unoccupied(idle=free) crane for this vessel

for (craneIdx = 0; craneIdx < num\_of\_cranes; craneIdx++)

if (cranesVessleIDs[craneIdx] == CRANE\_AVAILABLE)

{

cranesVessleIDs[craneIdx] = vessel\_ID; //update cell with the vessel id

cranesWithVessels++; //we need to know when all cranes have finished

break; // done searching

}

Sleep(randSleepTime()); //simulate time to enter unloading quay

sprintf(strBuffer, "Vessel %d - settled down at Crane %d", vessel\_ID, craneIdx + 1);

consolePrint(strBuffer);

cargoWeights[craneIdx] = rand() % (MAX\_CARGO\_WEIGHT - MIN\_CARGO\_WEIGHT) + MIN\_CARGO\_WEIGHT;

sprintf(strBuffer, "Vessel %d - weight is %d tons", vessel\_ID, cargoWeights[craneIdx]);

consolePrint(strBuffer);

if (!ReleaseMutex(mutexAdt)) {

exitWithError("UnloadingQuay::Unexpected error mutex.V()", MUTEX\_RELEASE\_ERROR);

}

/\* Exit critical section \*/

// Let the crane (the consumer) know it has a job to do now

if (!ReleaseSemaphore(cranesInSemaphores[craneIdx], 1, NULL))

exitWithError("UnloadingQuay::Unexpected error cranesInSemaphores.V()", MUTEX\_RELEASE\_ERROR);

/\*

Another confusing point:

Releasing the crane (in order to start unloading the vessel cargo

can be be done after releasing the adt mutex but it is just because

the crane's thread work doesn't affect the Quay ADT state inside it,

Even though the cranes are part of the quay ADT.

No other vessel can enter the loading quay and settle at this crane

until current vessel will be released from it. so we are safe to do it

This design would probabely work for the case of each crane handling multiple vessels simultaniously

(The order of execution won't be guranteed)

\*/

return craneIdx;

}

//Departure of a vessel from the unloading quay and from a specific crane.

// (After crane has done unloading its cargo)

void leaveQuay(int vessel\_ID, int craneIdx)

{

char strBuffer[OUTPUT\_BUFFER\_LEN];

Sleep(randSleepTime()); //for simulating exit from Quay ADT

///////////////////////////////////////////////////////////////////////////

WaitForSingleObject(mutexAdt, INFINITE); //we are accessing the quay adt structure

cranesVessleIDs[craneIdx] = CRANE\_AVAILABLE;

cranesWithVessels--;

cargoWeights[craneIdx] = 0; //reset

sprintf(strBuffer, "Vessel %d - left Crane %d and Exiting the UnloadingQuay", vessel\_ID, craneIdx + 1);

consolePrint(strBuffer);

if (cranesWithVessels == 0)

{

isQuayADTFree = true;

//////////////////////////////////////////////////

WaitForSingleObject(barrierMutex, INFINITE);

if (vesselsInBarrier >= num\_of\_cranes)

{

releaseFromBarrier();

}

if (!ReleaseMutex(barrierMutex))

{

exitWithError("UnloadingQuay::Unexpected error when releasing barrierMutex", -1);

}

//////////////////////////////////////////////////

}

if (!ReleaseMutex(mutexAdt))

{

exitWithError("UnloadingQuay::Unexpected error when releasing ADT mutex", -1);

}

///////////////////////////////////////////////////////////////////////////

}

// release M=num\_of\_cranes vessels from the head of barrier queue

void releaseFromBarrier()

{

char strBuffer[OUTPUT\_BUFFER\_LEN];

sprintf(strBuffer, "EilatPort: Barrier is now releasing %d Vessels", num\_of\_cranes);

consolePrint(strBuffer);

// Allow the num\_of\_cranes vessels from head of queue to go to the unloading quay

// (pops a vessel from the barrier queue head)

for (int i = 0; i < num\_of\_cranes; i++)

{

if (!ReleaseSemaphore(barrierQueue[barrierHeadIdx++], 1, NULL)) {

sprintf(strBuffer, "releaseFromBarrier:: Unexpected error barrierQueue[%d].V()\n", barrierHeadIdx - 1);

exitWithError(strBuffer, SEMAPHORE\_RELEASE\_ERROR);

}

}

isQuayADTFree = false; //we have just sent vessels into the unloading quay ("ADT")

}

//Function to allocated & Initialise memory for static variables

// prior to starting Vessels Threads (Crane threads are created here).

void initStaticVars() {

writePipeMutex = CreateMutex(NULL, FALSE, NULL);

if (writePipeMutex == NULL)

{

exitWithError("EilatPort: Unexpected Error in writePipeMutex creation", PIPE\_USAGE\_ERROR);

}

barrierMutex = CreateMutex(NULL, FALSE, NULL);

if (barrierMutex == NULL)

{

exitWithError("EilatPort:Unexpected Error in barrierMutex Creation", MUTEX\_CREATION\_ERROR);

}

mutexAdt = CreateMutex(NULL, FALSE, NULL);

if (mutexAdt == NULL)

{

exitWithError("EilatPort:Unexpected Error in mutexAdt Creation", MUTEX\_CREATION\_ERROR);

}

vesselThreadsIDArr = (int\*)malloc(numOfVessels \* sizeof(int));

if (!vesselThreadsIDArr)

{

exitWithError("EilatPort: Error allocating vesselThreadsIDArr", MEMORY\_ALLOCATION\_ERROR);

}

cargoWeights = (int\*)malloc(num\_of\_cranes \* sizeof(int));

if (cargoWeights == NULL)

{

exitWithError("EilatPort: Error allocating cargoWeights", MEMORY\_ALLOCATION\_ERROR);

}

vesselThreads = (HANDLE\*)HeapAlloc(GetProcessHeap(), HEAP\_ZERO\_MEMORY, (numOfVessels \* sizeof(HANDLE)));

if (vesselThreads == NULL)

{

exitWithError("EilatPort: Error allocating Vessels threads", MEMORY\_ALLOCATION\_ERROR);

}

cranesInSemaphores = (HANDLE\*)HeapAlloc(GetProcessHeap(), HEAP\_ZERO\_MEMORY, (num\_of\_cranes \* sizeof(HANDLE)));

if (cranesInSemaphores == NULL)

{

exitWithError("EilatPort: Error creating cranesInSemaphores", MEMORY\_ALLOCATION\_ERROR);

}

cranesOutSemaphores = (HANDLE\*)HeapAlloc(GetProcessHeap(), HEAP\_ZERO\_MEMORY, (num\_of\_cranes \* sizeof(HANDLE)));

if (cranesOutSemaphores == NULL)

{

exitWithError("EilatPort: Error creating cranesOutSemaphores", MEMORY\_ALLOCATION\_ERROR);

}

barrierQueue = HeapAlloc(GetProcessHeap(), HEAP\_ZERO\_MEMORY, (numOfVessels \* sizeof(HANDLE)));

if (barrierQueue == NULL)

{

exitWithError("EilatPort: Error allocating barrierQueueArray", MEMORY\_ALLOCATION\_ERROR);

}

Cranes = (HANDLE\*)HeapAlloc(GetProcessHeap(), HEAP\_ZERO\_MEMORY, (num\_of\_cranes \* sizeof(HANDLE)));

if (Cranes == NULL)

{

exitWithError("EilatPort: Error allocating Cranes", MEMORY\_ALLOCATION\_ERROR);

}

craneIDsArr = (int\*)malloc(num\_of\_cranes \* sizeof(int));

if (craneIDsArr == NULL)

{

exitWithError("EilatPort: Error allocating Cranes IDs", MEMORY\_ALLOCATION\_ERROR);

}

cranesVessleIDs = (int\*)malloc(num\_of\_cranes \* sizeof(int));

if (cranesVessleIDs == NULL)

{

exitWithError("EilatPort: Error allocating Cranes array", MEMORY\_ALLOCATION\_ERROR);

}

for (int i = 0; i < num\_of\_cranes; i++)

{

cargoWeights[i] = 0;

cranesVessleIDs[i] = CRANE\_AVAILABLE;

craneIDsArr[i] = i + 1; //unique sequence number starting from 1

Cranes[i] = CreateThread(NULL, 0, Crane, &craneIDsArr[i], 0, &ThreadId);

if (!Cranes[i])

{

char formattedStringBuff[OUTPUT\_BUFFER\_LEN];

sprintf(formattedStringBuff, "EilatPort: Failed to create a thread for Crane %d", i);

exitWithError(formattedStringBuff, THREAD\_CREATION\_ERROR);

}

cranesInSemaphores[i] = CreateSemaphore(NULL, 0, 1, NULL);

cranesOutSemaphores[i] = CreateSemaphore(NULL, 0, 1, NULL);

}

for (int i = 0; i < numOfVessels; i++)

{

barrierQueue[i] = CreateSemaphore(NULL, 0, 1, NULL);;

if (!barrierQueue[i])

{

exitWithError("EilatPort: Error when creating mutex for a barrier", SEMAPHORE\_CREATION\_ERROR);

}

}

return;

}

//Function to clean global data and closing Threads after all Threads finish.

// It will be called only after both of these happend sequentialy:

// 1. All vessels in Eilat Port Finished their job

// 2. Unloading Quay Cranes have finished their life-cycle

void releaseResources()

{

CloseHandle(mutexAdt);

CloseHandle(barrierMutex);

CloseHandle(writePipeMutex);

CloseHandle(sharedPrintMutex);

for (int i = 0; i < numOfVessels; i++)

{

CloseHandle(barrierQueue[i]);

}

for (int i = 0; i < num\_of\_cranes; i++)

{

CloseHandle(cranesInSemaphores[i]);

CloseHandle(cranesOutSemaphores[i]);

CloseHandle(Cranes[i]); // Cranes end their life only when main thread is done

}

for (int i = 0; i < numOfVessels; i++)

{

CloseHandle(vesselThreads[i]);

}

free(cargoWeights);

free(cranesVessleIDs);

free(vesselThreadsIDArr);

free(craneIDsArr);

HeapFree(GetProcessHeap(), HEAP\_NO\_SERIALIZE, barrierQueue);

HeapFree(GetProcessHeap(), HEAP\_NO\_SERIALIZE, vesselThreads);

HeapFree(GetProcessHeap(), HEAP\_NO\_SERIALIZE, Cranes);

HeapFree(GetProcessHeap(), HEAP\_NO\_SERIALIZE, cranesInSemaphores);

HeapFree(GetProcessHeap(), HEAP\_NO\_SERIALIZE, cranesOutSemaphores);

}

// For synchronization between print requests (among both HaifaPort & EilatPort)

void consolePrint(char\* msg)

{

WaitForSingleObject(sharedPrintMutex, INFINITE);

printToLog(msg);

if (!ReleaseMutex(sharedPrintMutex)) {

printf("PrintToConsole::Unexpected error mutex.V()\n"); //

releaseResources();

exit(MUTEX\_RELEASE\_ERROR);

}

}

//The main thread is listening/waiting for numOfVessels messages from Haifa

// (on an Anonymous pipe) and creating a Vessel thread for each of them ("Eilat vessel").

void receiveVesselsFromHaifa()

{

int vessel\_ID;

int vesselsVisitedEilat = 0;

char strBuffer[OUTPUT\_BUFFER\_LEN];

while (vesselsVisitedEilat < numOfVessels)

{

if (!ReadFile(ReadHandle, readPipeBuffer, PIPE\_BUF\_LEN, &bytesRecieved, NULL))

{

exitWithError("Eilat ::Error Reading from pipe", PIPE\_USAGE\_ERROR);

}

sprintf(strBuffer, "Vessel %s - arrived @ Eilat Port ", readPipeBuffer);

consolePrint(strBuffer);

vessel\_ID = atoi(readPipeBuffer);

vesselThreadsIDArr[vessel\_ID - 1] = vessel\_ID;

vesselsVisitedEilat++;

//Create a thread to represent a vessel that is now in EilatPort

// The thread will work "in parallel" to the main thread

vesselThreads[vessel\_ID - 1] = CreateThread(NULL, 0, Vessel, &vesselThreadsIDArr[vessel\_ID - 1], 0, &ThreadId);

}

}

// Wait for all Cranes & Vessels Threads to finish

void waitForCleanUpCondition()

{

WaitForMultipleObjects(numOfVessels, vesselThreads, TRUE, INFINITE);

consolePrint("Eilat Port: All Vessel Threads in Eilat are done ");

// Wait for all Crane threads to finish

WaitForMultipleObjects(num\_of\_cranes, Cranes, TRUE, INFINITE);

consolePrint("Eilat Port: All Crane Threads are done ");

return;

}

//Checking the correctness of the number of vessels and returning a response to Haifa Process.

void checkNumOfVessels()

{

char tempWriteBuffer[PIPE\_BUF\_LEN];

bool is\_prime = isPrime(numOfVessels);

if (!is\_prime)

{

strcpy(tempWriteBuffer, VESSELS\_ALLOWED);

}

else

{

strcpy(tempWriteBuffer, VESSELS\_NOT\_ALLOWED);

}

//Write to the pipe

if (!WriteFile(WriteHandle, tempWriteBuffer, PIPE\_BUF\_LEN, &bytesSent, NULL))

exitWithError("EilatPort: Error writing to pipe", PIPE\_USAGE\_ERROR);

if (is\_prime)

exitWithError("EilatPort: number of vessels argument is prime!", NUM\_VESSELS\_ERROR);

return;

}

//Initialization of pipes handles at Eilat side

void setPipesHandles()

{

ReadHandle = GetStdHandle(STD\_INPUT\_HANDLE); // get the read handle of pipe 1

WriteHandle = GetStdHandle(STD\_OUTPUT\_HANDLE); // get the write handle of pipe 2

if ((WriteHandle == INVALID\_HANDLE\_VALUE) || (ReadHandle == INVALID\_HANDLE\_VALUE))

exitWithError("Eilat:: Unexpected error while setting pipes handles", PIPE\_HANDLE\_ERROR);

}

void logVesselAtBarrier(int vessel\_ID)

{

char strBuffer[OUTPUT\_BUFFER\_LEN];

sprintf(strBuffer, "Vessel %d - Entered the Barrier", vessel\_ID);

consolePrint(strBuffer);

}

int main(int argc, char\* argv[])

{

//TODO: can you make this work as a printing method to the parent process console?

//AttachConsole(ATTACH\_PARENT\_PROCESS);

//printf("\n \*\* This is a message from the child process. \*\* \n");

char strBuffer[OUTPUT\_BUFFER\_LEN];

srand((unsigned int)time(NULL)); //initialize the PRNG seed with pc current time

setPipesHandles();

// since we are printing to a common console - we must

// synchronize the printings since the first one for this process

sharedPrintMutex = OpenMutex(MUTEX\_ALL\_ACCESS, FALSE, TEXT(PRINT\_MUTEX));

if (!sharedPrintMutex) {

fprintf(stderr, "EilatPort: Error creating mutexPipe");

exit(OPEN\_MUTEX\_ERROR);

}

// Read num of vessels HaifaPort is asking to send from the relevant pipe

if (ReadFile(ReadHandle, readPipeBuffer, PIPE\_BUF\_LEN, &bytesRecieved, NULL)) {

sprintf(strBuffer, "EilatPort: Received request message of - %s - Vessels", readPipeBuffer);

consolePrint(strBuffer);

}

else {

exitWithError("EilatPort: Error reading from pipe", PIPE\_USAGE\_ERROR);

}

numOfVessels = atoi(readPipeBuffer);

checkNumOfVessels();

num\_of\_cranes = generateNumberOfCranes(numOfVessels);

sprintf(strBuffer, "EilatPort: Number of cranes is %d", num\_of\_cranes);

consolePrint(strBuffer);

initStaticVars();

receiveVesselsFromHaifa();

waitForCleanUpCondition();

consolePrint("Eilat Port: Now Exiting... ");

releaseResources();

return 0;

}

דוגמאות מייצגות לפלט ממסך ההרצה:

Num of vessels = 3:

[12:23:35] HaifaPort: Number of vessels is 3

[12:23:35] HaifaPort: EilatPort process has been created

[12:23:35] HaifaPort: Requesting permission from EilatPort to pass vessels

[12:23:35] EilatPort: Received request message of - 3 - Vessels

[12:23:35] Error - EilatPort: number of vessels argument is prime! (0)

[12:23:35] Error - HaifaPort: request wasn't approved, number of vessels invalid! (0)

Num of vessels = 6:

[12:20:17] HaifaPort: Number of vessels is 6

[12:20:17] HaifaPort: EilatPort process has been created

[12:20:17] HaifaPort: Requesting permission from EilatPort to pass vessels

[12:20:17] EilatPort: Received request message of - 6 - Vessels

[12:20:17] EilatPort: Number of cranes is 2

[12:20:17] HaifaPort: request to pass was approved by EilatPort

[12:20:17] Crane - 1 has been created

[12:20:17] Crane - 2 has been created

[12:20:17] Vessel 4 - starts sailing @ Haifa Port

[12:20:17] Vessel 2 - starts sailing @ Haifa Port

[12:20:17] Vessel 3 - starts sailing @ Haifa Port

[12:20:17] Vessel 1 - starts sailing @ Haifa Port

[12:20:17] Vessel 6 - starts sailing @ Haifa Port

[12:20:17] Vessel 5 - starts sailing @ Haifa Port

[12:20:17] Vessel 4 - entering Canal: Med. Sea ==> Red Sea

[12:20:17] Vessel 4 - arrived @ Eilat Port

[12:20:17] Vessel 5 - entering Canal: Med. Sea ==> Red Sea

[12:20:17] Vessel 4 - Entered the Barrier

[12:20:18] Vessel 5 - arrived @ Eilat Port

[12:20:18] Vessel 6 - entering Canal: Med. Sea ==> Red Sea

[12:20:18] Vessel 5 - Entered the Barrier

[12:20:18] EilatPort: Barrier is now releasing 2 Vessels

[12:20:18] Vessel 5 - Released from barrier to unloading quay

[12:20:18] Vessel 4 - Released from barrier to unloading quay

[12:20:18] Vessel 3 - entering Canal: Med. Sea ==> Red Sea

[12:20:18] Vessel 6 - arrived @ Eilat Port

[12:20:18] Vessel 4 - settled down at Crane 1

[12:20:18] Vessel 4 - weight is 39 tons

[12:20:18] vessel 4 - Starts unloading  39 Tons at Crane 1...

[12:20:19] vessel 4 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:20:19] Vessel 3 - arrived @ Eilat Port

[12:20:19] Vessel 1 - entering Canal: Med. Sea ==> Red Sea

[12:20:19] Vessel 5 - settled down at Crane 2

[12:20:19] Vessel 5 - weight is 39 tons

[12:20:19] vessel 5 - Starts unloading  39 Tons at Crane 2...

[12:20:19] Vessel 6 - Entered the Barrier

[12:20:19] Vessel 3 - Entered the Barrier

[12:20:19] vessel 5 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:20:19] Vessel 1 - arrived @ Eilat Port

[12:20:19] Vessel 2 - entering Canal: Med. Sea ==> Red Sea

[12:20:19] Vessel 1 - Entered the Barrier

[12:20:20] Vessel 2 - arrived @ Eilat Port

[12:20:20] Vessel 2 - Entered the Barrier

[12:20:21] Vessel 4 - left Crane 1 and Exiting the UnloadingQuay

[12:20:21] Vessel 4 - entering Canal: Red Sea ==> Med. Sea

[12:20:22] Vessel 5 - left Crane 2 and Exiting the UnloadingQuay

[12:20:22] EilatPort: Barrier is now releasing 2 Vessels

[12:20:22] Vessel 3 - Released from barrier to unloading quay

[12:20:22] Vessel 6 - Released from barrier to unloading quay

[12:20:22] Vessel 5 - entering Canal: Red Sea ==> Med. Sea

[12:20:22] Vessel 6 - settled down at Crane 1

[12:20:22] Vessel 6 - weight is 39 tons

[12:20:22] vessel 6 - Starts unloading  39 Tons at Crane 1...

[12:20:22] Vessel 4 - exiting Canal: Red Sea ==> Med. Sea

[12:20:22] Vessel 4 - done sailing @ Haifa Port

[12:20:23] vessel 6 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:20:23] Vessel 3 - settled down at Crane 2

[12:20:23] Vessel 3 - weight is 39 tons

[12:20:23] vessel 3 - Starts unloading  39 Tons at Crane 2...

[12:20:23] Vessel 5 - exiting Canal: Red Sea ==> Med. Sea

[12:20:23] Vessel 5 - done sailing @ Haifa Port

[12:20:23] vessel 3 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:20:25] Vessel 6 - left Crane 1 and Exiting the UnloadingQuay

[12:20:25] Vessel 6 - entering Canal: Red Sea ==> Med. Sea

[12:20:26] Vessel 3 - left Crane 2 and Exiting the UnloadingQuay

[12:20:26] EilatPort: Barrier is now releasing 2 Vessels

[12:20:26] Vessel 3 - entering Canal: Red Sea ==> Med. Sea

[12:20:26] Vessel 2 - Released from barrier to unloading quay

[12:20:26] Vessel 1 - Released from barrier to unloading quay

[12:20:26] Vessel 1 - settled down at Crane 1

[12:20:26] Vessel 1 - weight is 39 tons

[12:20:26] vessel 1 - Starts unloading  39 Tons at Crane 1...

[12:20:27] Vessel 6 - exiting Canal: Red Sea ==> Med. Sea

[12:20:27] Vessel 6 - done sailing @ Haifa Port

[12:20:27] vessel 1 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:20:27] Vessel 2 - settled down at Crane 2

[12:20:27] Vessel 2 - weight is 39 tons

[12:20:27] vessel 2 - Starts unloading  39 Tons at Crane 2...

[12:20:27] Vessel 3 - exiting Canal: Red Sea ==> Med. Sea

[12:20:27] Vessel 3 - done sailing @ Haifa Port

[12:20:27] vessel 2 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:20:29] Vessel 1 - left Crane 1 and Exiting the UnloadingQuay

[12:20:29] Vessel 1 - entering Canal: Red Sea ==> Med. Sea

[12:20:30] Vessel 2 - left Crane 2 and Exiting the UnloadingQuay

[12:20:30] Vessel 2 - entering Canal: Red Sea ==> Med. Sea

[12:20:31] Vessel 1 - exiting Canal: Red Sea ==> Med. Sea

[12:20:31] Vessel 1 - done sailing @ Haifa Port

[12:20:31] Vessel 2 - exiting Canal: Red Sea ==> Med. Sea

[12:20:31] Eilat Port:  All Vessel Threads in Eilat are done

[12:20:31] Vessel 2 - done sailing @ Haifa Port

[12:20:31] Eilat Port: All Crane Threads are done

[12:20:31] Eilat Port: Now Exiting...

[12:20:31] Haifa Port: All Vessel Threads are done

[12:20:31] Haifa Port: Haifa Port: Exiting...

Number of Vessels =50 (MAX) [a good candidate test case to help us spot race-condition bugs in implementation] :

[12:26:50] HaifaPort: Number of vessels is 50

[12:26:50] HaifaPort: EilatPort process has been created

[12:26:50] HaifaPort: Requesting permission from EilatPort to pass vessels

[12:26:50] EilatPort: Received request message of - 50 - Vessels

[12:26:50] EilatPort: Number of cranes is 2

[12:26:50] HaifaPort: request to pass was approved by EilatPort

[12:26:50] Crane - 1 has been created

[12:26:50] Crane - 2 has been created

[12:26:50] Vessel 2 - starts sailing @ Haifa Port

[12:26:50] Vessel 3 - starts sailing @ Haifa Port

[12:26:50] Vessel 1 - starts sailing @ Haifa Port

[12:26:50] Vessel 4 - starts sailing @ Haifa Port

[12:26:50] Vessel 5 - starts sailing @ Haifa Port

[12:26:50] Vessel 6 - starts sailing @ Haifa Port

[12:26:50] Vessel 7 - starts sailing @ Haifa Port

[12:26:50] Vessel 8 - starts sailing @ Haifa Port

[12:26:50] Vessel 9 - starts sailing @ Haifa Port

[12:26:50] Vessel 10 - starts sailing @ Haifa Port

[12:26:50] Vessel 16 - starts sailing @ Haifa Port

[12:26:50] Vessel 12 - starts sailing @ Haifa Port

[12:26:50] Vessel 13 - starts sailing @ Haifa Port

[12:26:50] Vessel 14 - starts sailing @ Haifa Port

[12:26:50] Vessel 11 - starts sailing @ Haifa Port

[12:26:50] Vessel 15 - starts sailing @ Haifa Port

[12:26:50] Vessel 17 - starts sailing @ Haifa Port

[12:26:50] Vessel 18 - starts sailing @ Haifa Port

[12:26:50] Vessel 19 - starts sailing @ Haifa Port

[12:26:50] Vessel 20 - starts sailing @ Haifa Port

[12:26:50] Vessel 22 - starts sailing @ Haifa Port

[12:26:50] Vessel 26 - starts sailing @ Haifa Port

[12:26:50] Vessel 23 - starts sailing @ Haifa Port

[12:26:50] Vessel 24 - starts sailing @ Haifa Port

[12:26:50] Vessel 25 - starts sailing @ Haifa Port

[12:26:50] Vessel 27 - starts sailing @ Haifa Port

[12:26:50] Vessel 21 - starts sailing @ Haifa Port

[12:26:50] Vessel 29 - starts sailing @ Haifa Port

[12:26:50] Vessel 30 - starts sailing @ Haifa Port

[12:26:50] Vessel 28 - starts sailing @ Haifa Port

[12:26:50] Vessel 32 - starts sailing @ Haifa Port

[12:26:50] Vessel 31 - starts sailing @ Haifa Port

[12:26:50] Vessel 33 - starts sailing @ Haifa Port

[12:26:50] Vessel 34 - starts sailing @ Haifa Port

[12:26:50] Vessel 36 - starts sailing @ Haifa Port

[12:26:50] Vessel 37 - starts sailing @ Haifa Port

[12:26:50] Vessel 43 - starts sailing @ Haifa Port

[12:26:50] Vessel 38 - starts sailing @ Haifa Port

[12:26:50] Vessel 40 - starts sailing @ Haifa Port

[12:26:50] Vessel 39 - starts sailing @ Haifa Port

[12:26:50] Vessel 41 - starts sailing @ Haifa Port

[12:26:50] Vessel 42 - starts sailing @ Haifa Port

[12:26:50] Vessel 35 - starts sailing @ Haifa Port

[12:26:50] Vessel 45 - starts sailing @ Haifa Port

[12:26:50] Vessel 44 - starts sailing @ Haifa Port

[12:26:50] Vessel 46 - starts sailing @ Haifa Port

[12:26:50] Vessel 47 - starts sailing @ Haifa Port

[12:26:50] Vessel 50 - starts sailing @ Haifa Port

[12:26:50] Vessel 49 - starts sailing @ Haifa Port

[12:26:50] Vessel 48 - starts sailing @ Haifa Port

[12:26:50] Vessel 50 - entering Canal: Med. Sea ==> Red Sea

[12:26:50] Vessel 50 - arrived @ Eilat Port

[12:26:50] Vessel 49 - entering Canal: Med. Sea ==> Red Sea

[12:26:50] Vessel 50 - Entered the Barrier

[12:26:51] Vessel 49 - arrived @ Eilat Port

[12:26:51] Vessel 48 - entering Canal: Med. Sea ==> Red Sea

[12:26:51] Vessel 49 - Entered the Barrier

[12:26:51] EilatPort: Barrier is now releasing 2 Vessels

[12:26:51] Vessel 49 - Released from barrier to unloading quay

[12:26:51] Vessel 50 - Released from barrier to unloading quay

[12:26:51] Vessel 48 - arrived @ Eilat Port

[12:26:51] Vessel 47 - entering Canal: Med. Sea ==> Red Sea

[12:26:51] Vessel 50 - settled down at Crane 1

[12:26:51] Vessel 50 - weight is 39 tons

[12:26:51] vessel 50 - Starts unloading  39 Tons at Crane 1...

[12:26:51] vessel 50 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:26:52] Vessel 47 - arrived @ Eilat Port

[12:26:52] Vessel 46 - entering Canal: Med. Sea ==> Red Sea

[12:26:52] Vessel 49 - settled down at Crane 2

[12:26:52] Vessel 49 - weight is 39 tons

[12:26:52] Vessel 48 - Entered the Barrier

[12:26:52] vessel 49 - Starts unloading  39 Tons at Crane 2...

[12:26:52] Vessel 47 - Entered the Barrier

[12:26:52] vessel 49 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:26:52] Vessel 46 - arrived @ Eilat Port

[12:26:52] Vessel 44 - entering Canal: Med. Sea ==> Red Sea

[12:26:52] Vessel 46 - Entered the Barrier

[12:26:53] Vessel 44 - arrived @ Eilat Port

[12:26:53] Vessel 45 - entering Canal: Med. Sea ==> Red Sea

[12:26:53] Vessel 44 - Entered the Barrier

[12:26:53] Vessel 45 - arrived @ Eilat Port

[12:26:53] Vessel 35 - entering Canal: Med. Sea ==> Red Sea

[12:26:53] Vessel 45 - Entered the Barrier

[12:26:54] Vessel 35 - arrived @ Eilat Port

[12:26:54] Vessel 41 - entering Canal: Med. Sea ==> Red Sea

[12:26:54] Vessel 35 - Entered the Barrier

[12:26:54] Vessel 50 - left Crane 1 and Exiting the UnloadingQuay

[12:26:54] Vessel 50 - entering Canal: Red Sea ==> Med. Sea

[12:26:54] Vessel 41 - arrived @ Eilat Port

[12:26:54] Vessel 42 - entering Canal: Med. Sea ==> Red Sea

[12:26:54] Vessel 41 - Entered the Barrier

[12:26:55] Vessel 49 - left Crane 2 and Exiting the UnloadingQuay

[12:26:55] EilatPort: Barrier is now releasing 2 Vessels

[12:26:55] Vessel 48 - Released from barrier to unloading quay

[12:26:55] Vessel 47 - Released from barrier to unloading quay

[12:26:55] Vessel 49 - entering Canal: Red Sea ==> Med. Sea

[12:26:55] Vessel 42 - arrived @ Eilat Port

[12:26:55] Vessel 39 - entering Canal: Med. Sea ==> Red Sea

[12:26:55] Vessel 48 - settled down at Crane 1

[12:26:55] Vessel 48 - weight is 39 tons

[12:26:55] vessel 48 - Starts unloading  39 Tons at Crane 1...

[12:26:55] Vessel 50 - exiting Canal: Red Sea ==> Med. Sea

[12:26:55] Vessel 50 - done sailing @ Haifa Port

[12:26:56] Vessel 39 - arrived @ Eilat Port

[12:26:56] Vessel 40 - entering Canal: Med. Sea ==> Red Sea

[12:26:56] vessel 48 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:26:56] Vessel 47 - settled down at Crane 2

[12:26:56] Vessel 47 - weight is 39 tons

[12:26:56] vessel 47 - Starts unloading  39 Tons at Crane 2...

[12:26:56] Vessel 42 - Entered the Barrier

[12:26:56] Vessel 39 - Entered the Barrier

[12:26:56] Vessel 49 - exiting Canal: Red Sea ==> Med. Sea

[12:26:56] Vessel 49 - done sailing @ Haifa Port

[12:26:56] Vessel 40 - arrived @ Eilat Port

[12:26:56] Vessel 43 - entering Canal: Med. Sea ==> Red Sea

[12:26:56] Vessel 40 - Entered the Barrier

[12:26:56] vessel 47 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:26:57] Vessel 43 - arrived @ Eilat Port

[12:26:57] Vessel 38 - entering Canal: Med. Sea ==> Red Sea

[12:26:57] Vessel 43 - Entered the Barrier

[12:26:57] Vessel 38 - arrived @ Eilat Port

[12:26:57] Vessel 37 - entering Canal: Med. Sea ==> Red Sea

[12:26:57] Vessel 38 - Entered the Barrier

[12:26:58] Vessel 37 - arrived @ Eilat Port

[12:26:58] Vessel 36 - entering Canal: Med. Sea ==> Red Sea

[12:26:58] Vessel 37 - Entered the Barrier

[12:26:58] Vessel 36 - arrived @ Eilat Port

[12:26:58] Vessel 34 - entering Canal: Med. Sea ==> Red Sea

[12:26:58] Vessel 36 - Entered the Barrier

[12:26:58] Vessel 48 - left Crane 1 and Exiting the UnloadingQuay

[12:26:58] Vessel 48 - entering Canal: Red Sea ==> Med. Sea

[12:26:59] Vessel 34 - arrived @ Eilat Port

[12:26:59] Vessel 33 - entering Canal: Med. Sea ==> Red Sea

[12:26:59] Vessel 34 - Entered the Barrier

[12:26:59] Vessel 47 - left Crane 2 and Exiting the UnloadingQuay

[12:26:59] EilatPort: Barrier is now releasing 2 Vessels

[12:26:59] Vessel 47 - entering Canal: Red Sea ==> Med. Sea

[12:26:59] Vessel 46 - Released from barrier to unloading quay

[12:26:59] Vessel 44 - Released from barrier to unloading quay

[12:26:59] Vessel 31 - entering Canal: Med. Sea ==> Red Sea

[12:26:59] Vessel 33 - arrived @ Eilat Port

[12:26:59] Vessel 44 - settled down at Crane 1

[12:26:59] Vessel 44 - weight is 39 tons

[12:26:59] vessel 44 - Starts unloading  39 Tons at Crane 1...

[12:26:59] Vessel 48 - exiting Canal: Red Sea ==> Med. Sea

[12:26:59] Vessel 48 - done sailing @ Haifa Port

[12:27:00] Vessel 31 - arrived @ Eilat Port

[12:27:00] Vessel 32 - entering Canal: Med. Sea ==> Red Sea

[12:27:00] vessel 44 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:00] Vessel 46 - settled down at Crane 2

[12:27:00] Vessel 46 - weight is 39 tons

[12:27:00] vessel 46 - Starts unloading  39 Tons at Crane 2...

[12:27:00] Vessel 33 - Entered the Barrier

[12:27:00] Vessel 31 - Entered the Barrier

[12:27:00] Vessel 47 - exiting Canal: Red Sea ==> Med. Sea

[12:27:00] Vessel 47 - done sailing @ Haifa Port

[12:27:00] Vessel 32 - arrived @ Eilat Port

[12:27:00] Vessel 28 - entering Canal: Med. Sea ==> Red Sea

[12:27:00] Vessel 32 - Entered the Barrier

[12:27:00] vessel 46 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:27:01] Vessel 30 - entering Canal: Med. Sea ==> Red Sea

[12:27:01] Vessel 28 - arrived @ Eilat Port

[12:27:01] Vessel 28 - Entered the Barrier

[12:27:01] Vessel 30 - arrived @ Eilat Port

[12:27:01] Vessel 29 - entering Canal: Med. Sea ==> Red Sea

[12:27:01] Vessel 30 - Entered the Barrier

[12:27:02] Vessel 29 - arrived @ Eilat Port

[12:27:02] Vessel 21 - entering Canal: Med. Sea ==> Red Sea

[12:27:02] Vessel 29 - Entered the Barrier

[12:27:02] Vessel 21 - arrived @ Eilat Port

[12:27:02] Vessel 25 - entering Canal: Med. Sea ==> Red Sea

[12:27:02] Vessel 44 - left Crane 1 and Exiting the UnloadingQuay

[12:27:02] Vessel 44 - entering Canal: Red Sea ==> Med. Sea

[12:27:02] Vessel 21 - Entered the Barrier

[12:27:03] Vessel 46 - left Crane 2 and Exiting the UnloadingQuay

[12:27:03] Vessel 25 - arrived @ Eilat Port

[12:27:03] Vessel 24 - entering Canal: Med. Sea ==> Red Sea

[12:27:03] EilatPort: Barrier is now releasing 2 Vessels

[12:27:03] Vessel 46 - entering Canal: Red Sea ==> Med. Sea

[12:27:03] Vessel 45 - Released from barrier to unloading quay

[12:27:03] Vessel 35 - Released from barrier to unloading quay

[12:27:03] Vessel 25 - Entered the Barrier

[12:27:03] Vessel 24 - arrived @ Eilat Port

[12:27:03] Vessel 27 - entering Canal: Med. Sea ==> Red Sea

[12:27:03] Vessel 45 - settled down at Crane 1

[12:27:03] Vessel 45 - weight is 39 tons

[12:27:03] vessel 45 - Starts unloading  39 Tons at Crane 1...

[12:27:03] Vessel 44 - exiting Canal: Red Sea ==> Med. Sea

[12:27:03] Vessel 44 - done sailing @ Haifa Port

[12:27:04] Vessel 27 - arrived @ Eilat Port

[12:27:04] Vessel 26 - entering Canal: Med. Sea ==> Red Sea

[12:27:04] Vessel 35 - settled down at Crane 2

[12:27:04] Vessel 35 - weight is 39 tons

[12:27:04] Vessel 24 - Entered the Barrier

[12:27:04] vessel 35 - Starts unloading  39 Tons at Crane 2...

[12:27:04] Vessel 27 - Entered the Barrier

[12:27:04] Vessel 46 - exiting Canal: Red Sea ==> Med. Sea

[12:27:04] Vessel 46 - done sailing @ Haifa Port

[12:27:04] Vessel 26 - arrived @ Eilat Port

[12:27:04] Vessel 22 - entering Canal: Med. Sea ==> Red Sea

[12:27:04] Vessel 26 - Entered the Barrier

[12:27:05] Vessel 22 - arrived @ Eilat Port

[12:27:05] Vessel 20 - entering Canal: Med. Sea ==> Red Sea

[12:27:05] Vessel 22 - Entered the Barrier

[12:27:05] Vessel 20 - arrived @ Eilat Port

[12:27:05] Vessel 23 - entering Canal: Med. Sea ==> Red Sea

[12:27:05] Vessel 20 - Entered the Barrier

[12:27:06] Vessel 23 - arrived @ Eilat Port

[12:27:06] Vessel 19 - entering Canal: Med. Sea ==> Red Sea

[12:27:06] Vessel 23 - Entered the Barrier

[12:27:06] vessel 45 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:06] Vessel 19 - arrived @ Eilat Port

[12:27:06] Vessel 17 - entering Canal: Med. Sea ==> Red Sea

[12:27:06] Vessel 19 - Entered the Barrier

[12:27:06] vessel 35 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:27:07] Vessel 17 - arrived @ Eilat Port

[12:27:07] Vessel 18 - entering Canal: Med. Sea ==> Red Sea

[12:27:07] Vessel 17 - Entered the Barrier

[12:27:07] Vessel 18 - arrived @ Eilat Port

[12:27:07] Vessel 11 - entering Canal: Med. Sea ==> Red Sea

[12:27:07] Vessel 18 - Entered the Barrier

[12:27:08] Vessel 11 - arrived @ Eilat Port

[12:27:08] Vessel 14 - entering Canal: Med. Sea ==> Red Sea

[12:27:08] Vessel 11 - Entered the Barrier

[12:27:08] Vessel 14 - arrived @ Eilat Port

[12:27:08] Vessel 15 - entering Canal: Med. Sea ==> Red Sea

[12:27:08] Vessel 14 - Entered the Barrier

[12:27:08] Vessel 45 - left Crane 1 and Exiting the UnloadingQuay

[12:27:08] Vessel 45 - entering Canal: Red Sea ==> Med. Sea

[12:27:09] Vessel 15 - arrived @ Eilat Port

[12:27:09] Vessel 12 - entering Canal: Med. Sea ==> Red Sea

[12:27:09] Vessel 15 - Entered the Barrier

[12:27:09] Vessel 35 - left Crane 2 and Exiting the UnloadingQuay

[12:27:09] EilatPort: Barrier is now releasing 2 Vessels

[12:27:09] Vessel 42 - Released from barrier to unloading quay

[12:27:09] Vessel 41 - Released from barrier to unloading quay

[12:27:09] Vessel 35 - entering Canal: Red Sea ==> Med. Sea

[12:27:09] Vessel 12 - arrived @ Eilat Port

[12:27:09] Vessel 13 - entering Canal: Med. Sea ==> Red Sea

[12:27:09] Vessel 41 - settled down at Crane 1

[12:27:09] Vessel 41 - weight is 39 tons

[12:27:09] vessel 41 - Starts unloading  39 Tons at Crane 1...

[12:27:10] Vessel 45 - exiting Canal: Red Sea ==> Med. Sea

[12:27:10] Vessel 45 - done sailing @ Haifa Port

[12:27:10] Vessel 13 - arrived @ Eilat Port

[12:27:10] Vessel 16 - entering Canal: Med. Sea ==> Red Sea

[12:27:10] Vessel 42 - settled down at Crane 2

[12:27:10] Vessel 42 - weight is 39 tons

[12:27:10] Vessel 12 - Entered the Barrier

[12:27:10] vessel 42 - Starts unloading  39 Tons at Crane 2...

[12:27:10] Vessel 13 - Entered the Barrier

[12:27:10] Vessel 35 - exiting Canal: Red Sea ==> Med. Sea

[12:27:10] Vessel 35 - done sailing @ Haifa Port

[12:27:10] Vessel 16 - arrived @ Eilat Port

[12:27:10] Vessel 10 - entering Canal: Med. Sea ==> Red Sea

[12:27:10] Vessel 16 - Entered the Barrier

[12:27:11] vessel 41 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:11] Vessel 9 - entering Canal: Med. Sea ==> Red Sea

[12:27:11] Vessel 10 - arrived @ Eilat Port

[12:27:11] Vessel 10 - Entered the Barrier

[12:27:11] vessel 42 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:27:11] Vessel 9 - arrived @ Eilat Port

[12:27:11] Vessel 7 - entering Canal: Med. Sea ==> Red Sea

[12:27:11] Vessel 9 - Entered the Barrier

[12:27:12] Vessel 7 - arrived @ Eilat Port

[12:27:12] Vessel 8 - entering Canal: Med. Sea ==> Red Sea

[12:27:12] Vessel 7 - Entered the Barrier

[12:27:12] Vessel 8 - arrived @ Eilat Port

[12:27:12] Vessel 5 - entering Canal: Med. Sea ==> Red Sea

[12:27:12] Vessel 8 - Entered the Barrier

[12:27:13] Vessel 5 - arrived @ Eilat Port

[12:27:13] Vessel 6 - entering Canal: Med. Sea ==> Red Sea

[12:27:13] Vessel 5 - Entered the Barrier

[12:27:13] Vessel 41 - left Crane 1 and Exiting the UnloadingQuay

[12:27:13] Vessel 41 - entering Canal: Red Sea ==> Med. Sea

[12:27:13] Vessel 6 - arrived @ Eilat Port

[12:27:13] Vessel 1 - entering Canal: Med. Sea ==> Red Sea

[12:27:13] Vessel 6 - Entered the Barrier

[12:27:14] Vessel 42 - left Crane 2 and Exiting the UnloadingQuay

[12:27:14] EilatPort: Barrier is now releasing 2 Vessels

[12:27:14] Vessel 40 - Released from barrier to unloading quay

[12:27:14] Vessel 42 - entering Canal: Red Sea ==> Med. Sea

[12:27:14] Vessel 39 - Released from barrier to unloading quay

[12:27:14] Vessel 1 - arrived @ Eilat Port

[12:27:14] Vessel 3 - entering Canal: Med. Sea ==> Red Sea

[12:27:14] Vessel 40 - settled down at Crane 1

[12:27:14] Vessel 40 - weight is 39 tons

[12:27:14] vessel 40 - Starts unloading  39 Tons at Crane 1...

[12:27:14] Vessel 41 - exiting Canal: Red Sea ==> Med. Sea

[12:27:14] Vessel 41 - done sailing @ Haifa Port

[12:27:14] Vessel 3 - arrived @ Eilat Port

[12:27:14] Vessel 2 - entering Canal: Med. Sea ==> Red Sea

[12:27:15] Vessel 39 - settled down at Crane 2

[12:27:15] Vessel 39 - weight is 39 tons

[12:27:15] vessel 39 - Starts unloading  39 Tons at Crane 2...

[12:27:15] Vessel 1 - Entered the Barrier

[12:27:15] Vessel 3 - Entered the Barrier

[12:27:15] Vessel 2 - arrived @ Eilat Port

[12:27:15] Vessel 42 - exiting Canal: Red Sea ==> Med. Sea

[12:27:15] Vessel 4 - entering Canal: Med. Sea ==> Red Sea

[12:27:15] Vessel 2 - Entered the Barrier

[12:27:15] Vessel 42 - done sailing @ Haifa Port

[12:27:15] vessel 40 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:16] Vessel 4 - arrived @ Eilat Port

[12:27:16] Vessel 4 - Entered the Barrier

[12:27:16] vessel 39 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:27:18] Vessel 40 - left Crane 1 and Exiting the UnloadingQuay

[12:27:18] Vessel 40 - entering Canal: Red Sea ==> Med. Sea

[12:27:18] Vessel 39 - left Crane 2 and Exiting the UnloadingQuay

[12:27:18] EilatPort: Barrier is now releasing 2 Vessels

[12:27:18] Vessel 39 - entering Canal: Red Sea ==> Med. Sea

[12:27:18] Vessel 43 - Released from barrier to unloading quay

[12:27:18] Vessel 38 - Released from barrier to unloading quay

[12:27:19] Vessel 43 - settled down at Crane 1

[12:27:19] Vessel 43 - weight is 39 tons

[12:27:19] vessel 43 - Starts unloading  39 Tons at Crane 1...

[12:27:19] Vessel 40 - exiting Canal: Red Sea ==> Med. Sea

[12:27:19] Vessel 40 - done sailing @ Haifa Port

[12:27:19] Vessel 38 - settled down at Crane 2

[12:27:19] Vessel 38 - weight is 39 tons

[12:27:19] vessel 38 - Starts unloading  39 Tons at Crane 2...

[12:27:19] Vessel 39 - exiting Canal: Red Sea ==> Med. Sea

[12:27:19] Vessel 39 - done sailing @ Haifa Port

[12:27:21] vessel 43 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:22] vessel 38 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:27:24] Vessel 43 - left Crane 1 and Exiting the UnloadingQuay

[12:27:24] Vessel 43 - entering Canal: Red Sea ==> Med. Sea

[12:27:24] Vessel 38 - left Crane 2 and Exiting the UnloadingQuay

[12:27:24] EilatPort: Barrier is now releasing 2 Vessels

[12:27:24] Vessel 38 - entering Canal: Red Sea ==> Med. Sea

[12:27:24] Vessel 37 - Released from barrier to unloading quay

[12:27:24] Vessel 36 - Released from barrier to unloading quay

[12:27:25] Vessel 36 - settled down at Crane 1

[12:27:25] Vessel 36 - weight is 39 tons

[12:27:25] vessel 36 - Starts unloading  39 Tons at Crane 1...

[12:27:25] Vessel 43 - exiting Canal: Red Sea ==> Med. Sea

[12:27:25] Vessel 43 - done sailing @ Haifa Port

[12:27:25] Vessel 37 - settled down at Crane 2

[12:27:25] Vessel 37 - weight is 39 tons

[12:27:25] vessel 37 - Starts unloading  39 Tons at Crane 2...

[12:27:26] Vessel 38 - exiting Canal: Red Sea ==> Med. Sea

[12:27:26] Vessel 38 - done sailing @ Haifa Port

[12:27:27] vessel 36 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:28] vessel 37 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:27:30] Vessel 36 - left Crane 1 and Exiting the UnloadingQuay

[12:27:30] Vessel 36 - entering Canal: Red Sea ==> Med. Sea

[12:27:30] Vessel 37 - left Crane 2 and Exiting the UnloadingQuay

[12:27:30] EilatPort: Barrier is now releasing 2 Vessels

[12:27:30] Vessel 37 - entering Canal: Red Sea ==> Med. Sea

[12:27:30] Vessel 34 - Released from barrier to unloading quay

[12:27:30] Vessel 33 - Released from barrier to unloading quay

[12:27:31] Vessel 34 - settled down at Crane 1

[12:27:31] Vessel 34 - weight is 39 tons

[12:27:31] vessel 34 - Starts unloading  39 Tons at Crane 1...

[12:27:31] vessel 34 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:31] Vessel 36 - exiting Canal: Red Sea ==> Med. Sea

[12:27:31] Vessel 36 - done sailing @ Haifa Port

[12:27:31] Vessel 33 - settled down at Crane 2

[12:27:31] Vessel 33 - weight is 39 tons

[12:27:31] vessel 33 - Starts unloading  39 Tons at Crane 2...

[12:27:31] vessel 33 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:27:32] Vessel 37 - exiting Canal: Red Sea ==> Med. Sea

[12:27:32] Vessel 37 - done sailing @ Haifa Port

[12:27:34] Vessel 34 - left Crane 1 and Exiting the UnloadingQuay

[12:27:34] Vessel 34 - entering Canal: Red Sea ==> Med. Sea

[12:27:34] Vessel 33 - left Crane 2 and Exiting the UnloadingQuay

[12:27:34] EilatPort: Barrier is now releasing 2 Vessels

[12:27:34] Vessel 32 - Released from barrier to unloading quay

[12:27:34] Vessel 31 - Released from barrier to unloading quay

[12:27:34] Vessel 33 - entering Canal: Red Sea ==> Med. Sea

[12:27:35] Vessel 31 - settled down at Crane 1

[12:27:35] Vessel 31 - weight is 39 tons

[12:27:35] vessel 31 - Starts unloading  39 Tons at Crane 1...

[12:27:35] Vessel 34 - exiting Canal: Red Sea ==> Med. Sea

[12:27:35] Vessel 34 - done sailing @ Haifa Port

[12:27:35] Vessel 32 - settled down at Crane 2

[12:27:35] vessel 31 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:35] Vessel 32 - weight is 39 tons

[12:27:35] vessel 32 - Starts unloading  39 Tons at Crane 2...

[12:27:35] Vessel 33 - exiting Canal: Red Sea ==> Med. Sea

[12:27:35] Vessel 33 - done sailing @ Haifa Port

[12:27:36] vessel 32 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:27:38] Vessel 31 - left Crane 1 and Exiting the UnloadingQuay

[12:27:38] Vessel 31 - entering Canal: Red Sea ==> Med. Sea

[12:27:38] Vessel 32 - left Crane 2 and Exiting the UnloadingQuay

[12:27:38] EilatPort: Barrier is now releasing 2 Vessels

[12:27:38] Vessel 32 - entering Canal: Red Sea ==> Med. Sea

[12:27:38] Vessel 28 - Released from barrier to unloading quay

[12:27:38] Vessel 30 - Released from barrier to unloading quay

[12:27:39] Vessel 30 - settled down at Crane 1

[12:27:39] Vessel 30 - weight is 39 tons

[12:27:39] vessel 30 - Starts unloading  39 Tons at Crane 1...

[12:27:39] Vessel 31 - exiting Canal: Red Sea ==> Med. Sea

[12:27:39] Vessel 31 - done sailing @ Haifa Port

[12:27:39] Vessel 28 - settled down at Crane 2

[12:27:39] Vessel 28 - weight is 39 tons

[12:27:39] vessel 28 - Starts unloading  39 Tons at Crane 2...

[12:27:39] Vessel 32 - exiting Canal: Red Sea ==> Med. Sea

[12:27:39] Vessel 32 - done sailing @ Haifa Port

[12:27:41] vessel 30 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:42] vessel 28 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:27:44] Vessel 30 - left Crane 1 and Exiting the UnloadingQuay

[12:27:44] Vessel 30 - entering Canal: Red Sea ==> Med. Sea

[12:27:45] Vessel 28 - left Crane 2 and Exiting the UnloadingQuay

[12:27:45] EilatPort: Barrier is now releasing 2 Vessels

[12:27:45] Vessel 21 - Released from barrier to unloading quay

[12:27:45] Vessel 29 - Released from barrier to unloading quay

[12:27:45] Vessel 28 - entering Canal: Red Sea ==> Med. Sea

[12:27:45] Vessel 29 - settled down at Crane 1

[12:27:45] Vessel 29 - weight is 39 tons

[12:27:45] vessel 29 - Starts unloading  39 Tons at Crane 1...

[12:27:45] Vessel 30 - exiting Canal: Red Sea ==> Med. Sea

[12:27:45] Vessel 30 - done sailing @ Haifa Port

[12:27:46] Vessel 21 - settled down at Crane 2

[12:27:46] Vessel 21 - weight is 39 tons

[12:27:46] vessel 21 - Starts unloading  39 Tons at Crane 2...

[12:27:46] Vessel 28 - exiting Canal: Red Sea ==> Med. Sea

[12:27:46] Vessel 28 - done sailing @ Haifa Port

[12:27:46] vessel 29 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:47] vessel 21 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:27:49] Vessel 29 - left Crane 1 and Exiting the UnloadingQuay

[12:27:49] Vessel 29 - entering Canal: Red Sea ==> Med. Sea

[12:27:49] Vessel 21 - left Crane 2 and Exiting the UnloadingQuay

[12:27:49] EilatPort: Barrier is now releasing 2 Vessels

[12:27:49] Vessel 24 - Released from barrier to unloading quay

[12:27:49] Vessel 25 - Released from barrier to unloading quay

[12:27:49] Vessel 21 - entering Canal: Red Sea ==> Med. Sea

[12:27:50] Vessel 25 - settled down at Crane 1

[12:27:50] Vessel 25 - weight is 39 tons

[12:27:50] vessel 25 - Starts unloading  39 Tons at Crane 1...

[12:27:50] Vessel 29 - exiting Canal: Red Sea ==> Med. Sea

[12:27:50] Vessel 29 - done sailing @ Haifa Port

[12:27:51] Vessel 24 - settled down at Crane 2

[12:27:51] Vessel 24 - weight is 39 tons

[12:27:51] vessel 24 - Starts unloading  39 Tons at Crane 2...

[12:27:51] Vessel 21 - exiting Canal: Red Sea ==> Med. Sea

[12:27:51] Vessel 21 - done sailing @ Haifa Port

[12:27:52] vessel 25 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:53] vessel 24 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:27:55] Vessel 25 - left Crane 1 and Exiting the UnloadingQuay

[12:27:55] Vessel 25 - entering Canal: Red Sea ==> Med. Sea

[12:27:55] Vessel 24 - left Crane 2 and Exiting the UnloadingQuay

[12:27:55] EilatPort: Barrier is now releasing 2 Vessels

[12:27:55] Vessel 24 - entering Canal: Red Sea ==> Med. Sea

[12:27:55] Vessel 27 - Released from barrier to unloading quay

[12:27:55] Vessel 26 - Released from barrier to unloading quay

[12:27:56] Vessel 26 - settled down at Crane 1

[12:27:56] Vessel 26 - weight is 39 tons

[12:27:56] vessel 26 - Starts unloading  39 Tons at Crane 1...

[12:27:56] Vessel 25 - exiting Canal: Red Sea ==> Med. Sea

[12:27:56] Vessel 25 - done sailing @ Haifa Port

[12:27:56] Vessel 27 - settled down at Crane 2

[12:27:56] Vessel 27 - weight is 39 tons

[12:27:56] vessel 27 - Starts unloading  39 Tons at Crane 2...

[12:27:57] Vessel 24 - exiting Canal: Red Sea ==> Med. Sea

[12:27:57] Vessel 24 - done sailing @ Haifa Port

[12:27:58] vessel 26 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:27:58] vessel 27 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:00] Vessel 26 - left Crane 1 and Exiting the UnloadingQuay

[12:28:00] Vessel 26 - entering Canal: Red Sea ==> Med. Sea

[12:28:01] Vessel 27 - left Crane 2 and Exiting the UnloadingQuay

[12:28:01] EilatPort: Barrier is now releasing 2 Vessels

[12:28:01] Vessel 27 - entering Canal: Red Sea ==> Med. Sea

[12:28:01] Vessel 22 - Released from barrier to unloading quay

[12:28:01] Vessel 20 - Released from barrier to unloading quay

[12:28:01] Vessel 20 - settled down at Crane 1

[12:28:01] Vessel 20 - weight is 39 tons

[12:28:01] vessel 20 - Starts unloading  39 Tons at Crane 1...

[12:28:02] Vessel 26 - exiting Canal: Red Sea ==> Med. Sea

[12:28:02] Vessel 26 - done sailing @ Haifa Port

[12:28:02] Vessel 22 - settled down at Crane 2

[12:28:02] Vessel 22 - weight is 39 tons

[12:28:02] vessel 22 - Starts unloading  39 Tons at Crane 2...

[12:28:02] Vessel 27 - exiting Canal: Red Sea ==> Med. Sea

[12:28:02] Vessel 27 - done sailing @ Haifa Port

[12:28:02] vessel 20 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:28:03] vessel 22 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:05] Vessel 20 - left Crane 1 and Exiting the UnloadingQuay

[12:28:05] Vessel 20 - entering Canal: Red Sea ==> Med. Sea

[12:28:06] Vessel 22 - left Crane 2 and Exiting the UnloadingQuay

[12:28:06] EilatPort: Barrier is now releasing 2 Vessels

[12:28:06] Vessel 22 - entering Canal: Red Sea ==> Med. Sea

[12:28:06] Vessel 19 - Released from barrier to unloading quay

[12:28:06] Vessel 23 - Released from barrier to unloading quay

[12:28:06] Vessel 23 - settled down at Crane 1

[12:28:06] Vessel 23 - weight is 39 tons

[12:28:06] vessel 23 - Starts unloading  39 Tons at Crane 1...

[12:28:06] Vessel 20 - exiting Canal: Red Sea ==> Med. Sea

[12:28:06] Vessel 20 - done sailing @ Haifa Port

[12:28:07] Vessel 19 - settled down at Crane 2

[12:28:07] vessel 23 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:28:07] Vessel 19 - weight is 39 tons

[12:28:07] vessel 19 - Starts unloading  39 Tons at Crane 2...

[12:28:07] Vessel 22 - exiting Canal: Red Sea ==> Med. Sea

[12:28:07] Vessel 22 - done sailing @ Haifa Port

[12:28:07] vessel 19 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:09] Vessel 23 - left Crane 1 and Exiting the UnloadingQuay

[12:28:09] Vessel 23 - entering Canal: Red Sea ==> Med. Sea

[12:28:10] Vessel 19 - left Crane 2 and Exiting the UnloadingQuay

[12:28:10] EilatPort: Barrier is now releasing 2 Vessels

[12:28:10] Vessel 19 - entering Canal: Red Sea ==> Med. Sea

[12:28:10] Vessel 17 - Released from barrier to unloading quay

[12:28:10] Vessel 18 - Released from barrier to unloading quay

[12:28:10] Vessel 17 - settled down at Crane 1

[12:28:10] Vessel 17 - weight is 39 tons

[12:28:10] vessel 17 - Starts unloading  39 Tons at Crane 1...

[12:28:10] vessel 17 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:28:10] Vessel 23 - exiting Canal: Red Sea ==> Med. Sea

[12:28:10] Vessel 23 - done sailing @ Haifa Port

[12:28:11] Vessel 18 - settled down at Crane 2

[12:28:11] Vessel 18 - weight is 39 tons

[12:28:11] vessel 18 - Starts unloading  39 Tons at Crane 2...

[12:28:11] vessel 18 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:11] Vessel 19 - exiting Canal: Red Sea ==> Med. Sea

[12:28:11] Vessel 19 - done sailing @ Haifa Port

[12:28:13] Vessel 17 - left Crane 1 and Exiting the UnloadingQuay

[12:28:13] Vessel 17 - entering Canal: Red Sea ==> Med. Sea

[12:28:13] Vessel 18 - left Crane 2 and Exiting the UnloadingQuay

[12:28:13] EilatPort: Barrier is now releasing 2 Vessels

[12:28:13] Vessel 18 - entering Canal: Red Sea ==> Med. Sea

[12:28:13] Vessel 14 - Released from barrier to unloading quay

[12:28:13] Vessel 11 - Released from barrier to unloading quay

[12:28:14] Vessel 11 - settled down at Crane 1

[12:28:14] Vessel 11 - weight is 39 tons

[12:28:14] vessel 11 - Starts unloading  39 Tons at Crane 1...

[12:28:14] Vessel 17 - exiting Canal: Red Sea ==> Med. Sea

[12:28:14] Vessel 17 - done sailing @ Haifa Port

[12:28:14] Vessel 14 - settled down at Crane 2

[12:28:14] Vessel 14 - weight is 39 tons

[12:28:14] vessel 14 - Starts unloading  39 Tons at Crane 2...

[12:28:15] Vessel 18 - exiting Canal: Red Sea ==> Med. Sea

[12:28:15] Vessel 18 - done sailing @ Haifa Port

[12:28:17] vessel 11 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:28:17] vessel 14 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:19] Vessel 11 - left Crane 1 and Exiting the UnloadingQuay

[12:28:19] Vessel 11 - entering Canal: Red Sea ==> Med. Sea

[12:28:20] Vessel 14 - left Crane 2 and Exiting the UnloadingQuay

[12:28:20] EilatPort: Barrier is now releasing 2 Vessels

[12:28:20] Vessel 14 - entering Canal: Red Sea ==> Med. Sea

[12:28:20] Vessel 15 - Released from barrier to unloading quay

[12:28:20] Vessel 12 - Released from barrier to unloading quay

[12:28:21] Vessel 12 - settled down at Crane 1

[12:28:21] Vessel 12 - weight is 39 tons

[12:28:21] vessel 12 - Starts unloading  39 Tons at Crane 1...

[12:28:21] Vessel 11 - exiting Canal: Red Sea ==> Med. Sea

[12:28:21] Vessel 11 - done sailing @ Haifa Port

[12:28:21] Vessel 15 - settled down at Crane 2

[12:28:21] Vessel 15 - weight is 39 tons

[12:28:21] vessel 15 - Starts unloading  39 Tons at Crane 2...

[12:28:21] Vessel 14 - exiting Canal: Red Sea ==> Med. Sea

[12:28:21] Vessel 14 - done sailing @ Haifa Port

[12:28:22] vessel 12 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:28:23] vessel 15 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:25] Vessel 12 - left Crane 1 and Exiting the UnloadingQuay

[12:28:25] Vessel 12 - entering Canal: Red Sea ==> Med. Sea

[12:28:25] Vessel 15 - left Crane 2 and Exiting the UnloadingQuay

[12:28:25] EilatPort: Barrier is now releasing 2 Vessels

[12:28:25] Vessel 16 - Released from barrier to unloading quay

[12:28:25] Vessel 15 - entering Canal: Red Sea ==> Med. Sea

[12:28:25] Vessel 13 - Released from barrier to unloading quay

[12:28:26] Vessel 13 - settled down at Crane 1

[12:28:26] Vessel 13 - weight is 39 tons

[12:28:26] vessel 13 - Starts unloading  39 Tons at Crane 1...

[12:28:26] Vessel 12 - exiting Canal: Red Sea ==> Med. Sea

[12:28:26] Vessel 12 - done sailing @ Haifa Port

[12:28:27] Vessel 16 - settled down at Crane 2

[12:28:27] Vessel 16 - weight is 39 tons

[12:28:27] vessel 16 - Starts unloading  39 Tons at Crane 2...

[12:28:27] Vessel 15 - exiting Canal: Red Sea ==> Med. Sea

[12:28:27] Vessel 15 - done sailing @ Haifa Port

[12:28:28] vessel 13 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:28:29] vessel 16 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:31] Vessel 13 - left Crane 1 and Exiting the UnloadingQuay

[12:28:31] Vessel 13 - entering Canal: Red Sea ==> Med. Sea

[12:28:32] Vessel 16 - left Crane 2 and Exiting the UnloadingQuay

[12:28:32] EilatPort: Barrier is now releasing 2 Vessels

[12:28:32] Vessel 16 - entering Canal: Red Sea ==> Med. Sea

[12:28:32] Vessel 10 - Released from barrier to unloading quay

[12:28:32] Vessel 9 - Released from barrier to unloading quay

[12:28:32] Vessel 9 - settled down at Crane 1

[12:28:32] Vessel 9 - weight is 39 tons

[12:28:32] vessel 9 - Starts unloading  39 Tons at Crane 1...

[12:28:32] Vessel 13 - exiting Canal: Red Sea ==> Med. Sea

[12:28:32] Vessel 13 - done sailing @ Haifa Port

[12:28:33] Vessel 10 - settled down at Crane 2

[12:28:33] Vessel 10 - weight is 39 tons

[12:28:33] vessel 10 - Starts unloading  39 Tons at Crane 2...

[12:28:33] Vessel 16 - exiting Canal: Red Sea ==> Med. Sea

[12:28:33] Vessel 16 - done sailing @ Haifa Port

[12:28:35] vessel 9 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:28:35] vessel 10 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:37] Vessel 9 - left Crane 1 and Exiting the UnloadingQuay

[12:28:37] Vessel 9 - entering Canal: Red Sea ==> Med. Sea

[12:28:38] Vessel 10 - left Crane 2 and Exiting the UnloadingQuay

[12:28:38] EilatPort: Barrier is now releasing 2 Vessels

[12:28:38] Vessel 10 - entering Canal: Red Sea ==> Med. Sea

[12:28:38] Vessel 7 - Released from barrier to unloading quay

[12:28:38] Vessel 8 - Released from barrier to unloading quay

[12:28:38] Vessel 8 - settled down at Crane 1

[12:28:38] Vessel 8 - weight is 39 tons

[12:28:38] vessel 8 - Starts unloading  39 Tons at Crane 1...

[12:28:38] Vessel 9 - exiting Canal: Red Sea ==> Med. Sea

[12:28:38] Vessel 9 - done sailing @ Haifa Port

[12:28:39] Vessel 7 - settled down at Crane 2

[12:28:39] Vessel 7 - weight is 39 tons

[12:28:39] vessel 7 - Starts unloading  39 Tons at Crane 2...

[12:28:39] Vessel 10 - exiting Canal: Red Sea ==> Med. Sea

[12:28:39] Vessel 10 - done sailing @ Haifa Port

[12:28:41] vessel 8 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:28:41] vessel 7 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:43] Vessel 8 - left Crane 1 and Exiting the UnloadingQuay

[12:28:43] Vessel 8 - entering Canal: Red Sea ==> Med. Sea

[12:28:44] Vessel 7 - left Crane 2 and Exiting the UnloadingQuay

[12:28:44] EilatPort: Barrier is now releasing 2 Vessels

[12:28:44] Vessel 7 - entering Canal: Red Sea ==> Med. Sea

[12:28:44] Vessel 5 - Released from barrier to unloading quay

[12:28:44] Vessel 6 - Released from barrier to unloading quay

[12:28:44] Vessel 6 - settled down at Crane 1

[12:28:45] Vessel 6 - weight is 39 tons

[12:28:45] vessel 6 - Starts unloading  39 Tons at Crane 1...

[12:28:45] Vessel 8 - exiting Canal: Red Sea ==> Med. Sea

[12:28:45] Vessel 8 - done sailing @ Haifa Port

[12:28:45] Vessel 5 - settled down at Crane 2

[12:28:45] Vessel 5 - weight is 39 tons

[12:28:45] vessel 5 - Starts unloading  39 Tons at Crane 2...

[12:28:45] Vessel 7 - exiting Canal: Red Sea ==> Med. Sea

[12:28:45] Vessel 7 - done sailing @ Haifa Port

[12:28:45] vessel 6 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:28:46] vessel 5 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:48] Vessel 6 - left Crane 1 and Exiting the UnloadingQuay

[12:28:48] Vessel 6 - entering Canal: Red Sea ==> Med. Sea

[12:28:48] Vessel 5 - left Crane 2 and Exiting the UnloadingQuay

[12:28:48] EilatPort: Barrier is now releasing 2 Vessels

[12:28:48] Vessel 3 - Released from barrier to unloading quay

[12:28:48] Vessel 1 - Released from barrier to unloading quay

[12:28:48] Vessel 5 - entering Canal: Red Sea ==> Med. Sea

[12:28:49] Vessel 1 - settled down at Crane 1

[12:28:49] Vessel 1 - weight is 39 tons

[12:28:49] vessel 1 - Starts unloading  39 Tons at Crane 1...

[12:28:49] Vessel 6 - exiting Canal: Red Sea ==> Med. Sea

[12:28:49] Vessel 6 - done sailing @ Haifa Port

[12:28:49] vessel 1 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:28:50] Vessel 3 - settled down at Crane 2

[12:28:50] Vessel 3 - weight is 39 tons

[12:28:50] vessel 3 - Starts unloading  39 Tons at Crane 2...

[12:28:50] Vessel 5 - exiting Canal: Red Sea ==> Med. Sea

[12:28:50] Vessel 5 - done sailing @ Haifa Port

[12:28:50] vessel 3 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:52] Vessel 1 - left Crane 1 and Exiting the UnloadingQuay

[12:28:52] Vessel 1 - entering Canal: Red Sea ==> Med. Sea

[12:28:52] Vessel 3 - left Crane 2 and Exiting the UnloadingQuay

[12:28:52] EilatPort: Barrier is now releasing 2 Vessels

[12:28:52] Vessel 4 - Released from barrier to unloading quay

[12:28:52] Vessel 2 - Released from barrier to unloading quay

[12:28:52] Vessel 3 - entering Canal: Red Sea ==> Med. Sea

[12:28:53] Vessel 2 - settled down at Crane 1

[12:28:53] Vessel 2 - weight is 39 tons

[12:28:53] vessel 2 - Starts unloading  39 Tons at Crane 1...

[12:28:53] Vessel 1 - exiting Canal: Red Sea ==> Med. Sea

[12:28:53] Vessel 1 - done sailing @ Haifa Port

[12:28:53] vessel 2 - Finished unloading cargo of weight 39 Tons at Crane 1

[12:28:53] Vessel 4 - settled down at Crane 2

[12:28:53] Vessel 4 - weight is 39 tons

[12:28:53] vessel 4 - Starts unloading  39 Tons at Crane 2...

[12:28:54] Vessel 3 - exiting Canal: Red Sea ==> Med. Sea

[12:28:54] Vessel 3 - done sailing @ Haifa Port

[12:28:54] vessel 4 - Finished unloading cargo of weight 39 Tons at Crane 2

[12:28:56] Vessel 2 - left Crane 1 and Exiting the UnloadingQuay

[12:28:56] Vessel 2 - entering Canal: Red Sea ==> Med. Sea

[12:28:56] Vessel 4 - left Crane 2 and Exiting the UnloadingQuay

[12:28:56] Vessel 4 - entering Canal: Red Sea ==> Med. Sea

[12:28:57] Vessel 2 - exiting Canal: Red Sea ==> Med. Sea

[12:28:57] Vessel 2 - done sailing @ Haifa Port

[12:28:57] Vessel 4 - exiting Canal: Red Sea ==> Med. Sea

[12:28:57] Eilat Port:  All Vessel Threads in Eilat are done

[12:28:57] Vessel 4 - done sailing @ Haifa Port

[12:28:57] Eilat Port: All Crane Threads are done

[12:28:57] Eilat Port: Now Exiting...

[12:28:57] Haifa Port: All Vessel Threads are done

[12:28:57] Haifa Port: Haifa Port: Exiting...

נספח:

דרכים בהן ניתן למצוא באגים במימוש של סדר הפעולות כמו deadlock ו race condition (המתנה לסיום של מישהו אחר, מקביליות וכו'):

* לשנות במכוון את זמן ההמתנה של פעולות מסויימות לערך גדול מאוד כדי לראות שאכן יש המתנה לסיום של אותה פעולה והיא לא מתחילה לפני כן.

דרכים בהן ניתן לדבג את תהליך הבן בסביבת ויזואל סטודיו:

* לשתול הדפסות בקוד תהליך הבן – quick & dirty way בה נעשה שימוש

או לחפש באינטרנט דרכים/תוכנות מקצועיות לכך שהdebugger יעבור לעקוב אחרי תהליך הבן מרגע שנוצר

* <https://marketplace.visualstudio.com/items?itemName=vsdbgplat.MicrosoftChildProcessDebuggingPowerTool>
* X64dbg / WinDbg