**המשימה סנכרון בין תהליכים (threads) והתמודדות עם הודעות במספר תורים במערכת מקבילית**.

**דרישות מערכת - תיאור כללי**

המערכת היא יישום רב-תהליכים (multi-threaded) שמתזמן שליחת הודעות אקראיות למספר תהליכים, מאחסן את ההודעות בתורים נפרדים לכל תהליך, ומבצע השוואות תקופתיות בין התורים. כל תהליך פועל באופן עצמאי, אך מסונכרן עם יתר התהליכים באמצעות מנעולים (mutex) ומשתני תנאי (condition variables) , כאשר מערכת זו מתוכננת לפעול ברציפות ולבצע פעולות באופן מחזורי.

**שליחת הודעות אקראיות לתהליכים**

המערכת שולחת הודעות אקראיות לתהליכים, כאשר כל הודעה היא מספר שנבחר באופן אקראי מתוך טווח מוגדר מראש. תהליך שליחת ההודעות מתבצע על ידי התוכנית הראשית, ובכל סבב נבחרת אחת מתוך שלוש אפשרויות אקראיות:

1. שליחת הודעה לתהליך הראשון.
2. שליחת הודעה לתהליך השני.
3. שליחת הודעה לשני התהליכים במקביל.

לכל תהליך יש תור שבו הוא שומר את ההודעות שהוא מקבל, וההודעות נשלחות בקצב של אחת לשנייה. כלומר, יש להשהות את שליחת ההודעות במשך שנייה אחת בין כל הודעה והודעה.

**אחסון הודעות בתורים**

כל תהליך אחראי לאחסן את ההודעות שהוא מקבל בתור ייחודי לו. המערכת שומרת על שני תורים נפרדים:

* תור אחד עבור ההודעות שהתקבלו בתהליך הראשון.
* תור שני עבור ההודעות שהתקבלו בתהליך השני.

**השוואת הודעות בין תורים**

בכל חמש שניות, תהליך שלישי מבצע בדיקה של תורי ההודעות של התהליכים הראשון והשני. הוא משווה בין המספרים שנמצאים בראש כל תור:

* אם המספרים זהים, הם מוסרים משני התורים.
* אם המספר בתור הראשון קטן מהמספר בתור השני, המספר הקטן מוסר מהתור הראשון.
* אם המספר בתור השני קטן מהמספר בתור הראשון, המספר הקטן מוסר מהתור השני.

**סנכרון תהליכים**

הגישה לתורים חייבת להיות מסונכרנת בין כל התהליכים כדי למנוע מצבים של קריאה וכתיבה לא מבוקרת לתורים. לצורך כך, המערכת משתמשת במנעולים (mutex) ובמשתני תנאי (condition variables) שיאפשרו לתהליכים להמתין באופן אקטיבי להודעות חדשות ולהשתמש בתורים בצורה בטוחה.

המערכת חייבת לאפשר לתהליכים לפעול ברציפות מבלי להפריע זה לזה. כל תהליך יפעל רק כאשר יש לו הודעה חדשה שממתינה לעיבוד בתורו, כאשר תהליכי 1 ו-2 ימתינו להודעות מהתוכנית הראשית, ותהליך 3 ימתין לביצוע הבדיקה המחזורית.

**עדכון המשתמש**

המערכת צריכה לעדכן את המשתמש בכל פעם שמתבצעת פעולה כלשהי. לדוגמה, כאשר תהליך מקבל הודעה חדשה, יש להדפיס הודעה למסך המציינת את המספר שהתקבל ואת שם התהליך שקיבל אותו. כמו כן, במהלך הבדיקה המחזורית של תהליך 3, יש לעדכן את המשתמש במצב הנוכחי של התורים (גודל התור והמספרים בראש התור), ולציין אם נמצאו מספרים זהים או אם מספר כלשהו הוסר מאחד התורים.

**השהייה בין פעולות**

המערכת צריכה לבצע השהיות בין פעולות שונות:

1. השהייה של שנייה אחת בין כל שליחת הודעה מהתוכנית הראשית.
2. השהייה של חמש שניות בין כל בדיקה של תהליך 3 לצורך השוואת התורים.

**עבודה מתמשכת**

המערכת מתוכננת לפעול ברציפות ולבצע את הפעולות באופן מחזורי. התהליכים צריכים להמשיך לפעול לאורך זמן, לשלוח ולקבל הודעות באופן סדיר, ולבצע השוואות בין ההודעות שנמצאות בתורים בכל מחזור.

**ייצור מספרים אקראיים**

המערכת צריכה לייצר מספרים אקראיים בכל שליחת הודעה. המספרים ייבחרו מתוך טווחים מסוימים שנקבעים בהתאם להודעה האחרונה שנשלחה לתהליך הרלוונטי. לדוגמה, המספר שישלח לתהליך 1 צריך להיות גבוה יותר מהמספר האחרון שנשלח לתהליך זה, וכך גם לתהליך 2.

**יציבות המערכת**

המערכת צריכה להיות יציבה ולהימנע ממצבים שבהם גישה מקבילית לא מבוקרת לתורים עלולה לגרום לבעיות כמו קריסות או קונפליקטים בין תהליכים.

**הבדלים בין version1 לבין version2:**

1. **שימוש בפונקציה כללית לניהול הודעות**

// **version1**

// Two separate functions for thread 1 and thread 2:

void threadFunction1() {

// code

}

void threadFunction2() {

// code

}

// **version2**

// One general function that serves both threads:

void threadFunction (queue<int>& message\_queue, condition\_variable& cv, bool& new\_message\_flag,   
const string& thread\_name) {

// code

}

הפונקציה threadFunction היא דוגמה לפונקציה כללית שמשרתת שני תהליכים (threads) .  
הקלטים שלה הן מצביעים (references) למשתנים שונים, כך שהיא יכולה לשתף מידע בין התהליכים ולהגיב לשינויים בזמן אמת.

**קלטים של הפונקציה**

1. **queue<int>& message\_queue** **:**   
    תור שמכיל הודעות. השימוש בהפניה (reference) מאפשר לתהליך לגשת לתור המקורי ולא ליצור עותק שלו.
2. **condition\_variable& cv** **:**  
   משתנה שמטפל בהמתנות (blocking) של התהליכים. גם כאן, ההפניה משמעה שהשינויים משפיעים ישירות על ה-condition\_variable המקורי.
3. **bool& new\_message\_flag :**   
   דגל שמצביע אם יש הודעה חדשה. ההפניה מבטיחה שהשינויים ייראו בכל התהליכים.
4. **const string& thread\_name :**  
   שם התהליך, שמועבר כהפניה כדי לחסוך בזיכרון, אבל לא צפוי להשתנות.

**יתרונות השימוש בהפניה**

* + **שיתוף מידע בין תהליכים**:

כשאתה מעביר הפניה למשתנים, כל תהליך יכול לגשת ולשנות את אותם משתנים. זה מאפשר לתהליכים לשתף מידע בקלות.

1. **חיסכון בזיכרון**:

במקום ליצור עותקים של התורים או הדגלים (שיכולים להיות גדולים), אתה משתמש במידע המקורי. זה חוסך מקום בזיכרון ומפחית את העלות של יצירת עותקים.

1. **עדכון משתנים בזמן אמת**:

כל שינוי שנעשה על ידי תהליך אחד ייראה מיד על ידי תהליכים אחרים. זה מאוד חשוב במצבים בהם התהליכים תלויים זה בזה וצריכים לפעול על בסיס המידע המעודכן.

**הערה:**

הפונקציה threadFunction מתמודדת עם גישה משותפת למחסניות (תורים) מבלי להיתקל בבעיות גישה מתחרה  
 (race conditions) באמצעות שימוש במנעולים (mutex) ובמשתני תנאי (condition variables) .

1. **הסרת דגל הודעה עבור thread 3**

// **version1**

bool new\_message\_thread3 = false;

// **version2**

// bool new\_message\_thread3 = false;

1. **שיפוט על שם התהליך להדפסה**

// **version1**

// Prints with fixed name:

cout << "[Thread 1] Received number: " << number\_to\_send << endl;

// **version2**

// Prints with variable name:

cout << "[" << thread\_name << "] Received number: " << number\_to\_send << endl;