תרגיל 3 - סמסטר ב 2024-5.
סנכרון ושרת "קמור".

גירסא 1.01 - 1 ביוני 2025

בתרגיל נממש שרת על קבוצת נקודות בשריג דו מימדי אלגוריתם למציאת CONVEX HULL (או קמור).

שלב 1 - תכנית הממשת CH 10 נקודות

קלוט מהקלט הסטנדרטי מספר. +LF
המספר מתאר את מספר הנקודות

לאחר קריאת מספר הנקודות יקראו זוגות - כל זוג מייצג נקודה
(1,2) מייצג את הנקודה X=1 וY=2

אפשר להניח שכל הנקודות שנקראות הן FLOAT ושאין מידע נוסף בשורה פרט ל2 הנקודות ופסיק מפריד + סוף שורה. (LINE FEED)

לאחר קריאת הנקודות מצא את שטחו של הCONVEX HULL והחזר אותו
דוגמא לקלט

משמעות	שורה
4 נקודות	4
נקודה 0 0	0,0
נקודה 0 1	0,1
נקודה 1 1	1,1
נקודה 2 0	2,0

פלט

משמעות	שורה
שטח הקמור	1.5

שלב 2 - profiling - עשר נקודות

10 נקודות

בחן את מימוש CH. (שלב 1)
בחר שני ישומים של מבני נתונים או אלגוריתמים (לדוגמא למי שמשתמש בc++ מימוש של deque ומימוש של list)
בצע profiling לריצה של שלב 1 עם שני המימושים השונים. איזה מהמימושים יעיל יותר במקרה שלנו?

שלב 3 - קצת אינטרקציה - 10 נקודות

האלגוריתם שלנו יתמוך באינטרקציה עם לקוחות דרך stdin. כל פקודה תסתיים בLF.
(אפשר להניח שהקלט חוקי)

התוכנית תקבל עכשיו קלט מהstdin ותתמוך בפקודות הבאות

Newgraph n

קבל גרף חדש עם n נקודות חדשות - יש לקלוט m שורות של 2 מספרים (m נקודות כמו בסעיף הקודם)

CH

חשב את האלגוריתם של convex hull על הגרף הנוכחי הוצא פלט ל stdout

Newpoint i,j

הוסף נקודה חדשה

Removepoint i,j

מחק נקודה

*** אין פקודות אחרות

שלב 4 - ריבוי משתמשים - 10 נקודות

נמזג את התכנית של שלב 3 ואת הchat של beej. מטרתנו לשלב ביניהם כלומר נייצר שרת שיחזיק מבנה נתונים של גרף (**משותף לכל המשתמשים**) וכל משתמש מחובר יכול לייצר גרף חדש, להוסיף קשת ,להוריד קשת או להורות על חישוב CH על הגרף הנוכחי. ניתן להניח שהקלט חוקי.

פרוטוקול התקשורת בין השרת ללקוחות - יהיה טקסטואלי (כמו בשלב 3) על גבי port 9034 (הport של beej)

כל משתמש נותן פקודות (כפי שנתנו בשלב 3) או מורה על חישוב האלגוריתם.
משתמש שנותן פקודות שמשנות את הגרף צריך לחכות לשינוי או חישוב קודם שיסתיים.

* שים לב בסעיף זה עדיין לא צריך להגן על חישוב הגרף בעזרת mutex מכיוון שאנחנו עדין עובדים בטרד אחד ולכן לא יהיה שינוי וחישוב לגרף במקביל. (אנחנו גם לא נגן על יצירת גרף, למרות שיכולות היו להיות התנגשויות בשלב זה, נניח שהקלט חוקי גם בסדר)

שלב 5 - תבנית reactor - עשר נקודות

בנה ספרית תבניות עם תבנית ל reactor.
הreactor יכול לעבוד עם $select(2)$ או $poll(2)$ לבחירתכם.

הreactor יקבל fd ופונקציה לקרוא לה כאשר הfd חם

```
typedef (void *) (* reactorFunc) (int fd);
typedef reactor;

// starts new reactor and returns pointer to it
void * startReactor ();

// adds fd to Reactor (for reading) ; returns 0 on success.
Int  addFdToReactor(void * reactor, int fd, reactorFunc func); Int  addFdToReactor(void * reactor,
int fd, reactorFunc func);

// removes fd from reactor
Int removeFdFromReactor(void * reactor, int fd);

// stops reactor
int stopReactor(void * reactor);
```

אפשר להניח קלט חוקי (אם מוסיפים fd לreactor הוא לא קיים. אם מוציאים fd מהreactor הוא קיים)

Example

List fds;

```
reactorFuncs[MAX];

reactorFunction()
{
	for(;;) {
		Fdset reactorfds;
		Init reactorfds();
		Struct timeval tv;
		tv.tv_sec=1;
		tv.tv_usec=0;
		select(maxfd, &reactorfds, NULL, NULL,&tv);
		For (i=0 ; i < maxfd ; ++i) {
			If (FD_ISSET(i, reactorfds))
				pointersToFuncs[i](i);
		}
	}
}

Int  addFdToReactor(void * reactor, int fd, reactorFunc func)
{
	fds.insert(fd);
	reactorFuncs[fd]=func;
}

Int removeFdFromReactor(void * reactor, int fd)
{
	fd.delete(fd);
	rectorFuncs(fd)=NULL;
}
```

שלב 6 - מימוש שלב 4 בעזרת תבנית (שלב 5) - עשר נקודות

ממש את שלב 4 אבל בעזרת התבנית והספריה שמימשת בשלב 5.

שלב 7 - מימוש מרובה threads - עשר נקודות

חזור למימוש של שלב 4.
אנחנו לא מעוניינים בasync I/O יותר - במקום אנחנו מעוניינים בפתיחה של thread על כל לקוח חדש שנוצר.
(ככה שלשרת יש n+1 טרדים אם יש n לקוחות מחוברים. טרד שעושה accept ועוד n טרדים - אחד לכל לקוח).

שים לב שהגרף הוא משאב משותף ואנחנו צריכים להגן עליו בעזרת mutex. (כדי שלא ישתנה בזמן שטרד אחר עובד עליו - למשל מריץ את האלגוריתם).

שימו לב - המימוש של ה-mutex יכול להיות פשוט, אין צורך בהגנה נפרדת על כתיבות וקריאות.

שלב 8 - תבנית proactor - עשר נקודות

הוסף לספריה (שלב 5) תבנית של proactor.
התבנית תקבל socket עליו היא מאזינה. בכל פעם שמתחבר לקוח (עושים accept) הproactor יצור טרד חדש עם פונקציה שיקבל כפרמטר.

```
Typedef (void *) (* proactorFunc) (int sockfd)

// starts new proactor and returns proactor thread id.
pthread_t startProactor (int sockfd, proactorFunc threadFunc);
// stops proactor by threadid
int stopProactor(pthread_t tid);
```

שים לב שהגרף הוא משאב משותף ושינויים למשאב המשותף חייבים להיות מוגנים בmutex.
בנוסף לא יכול להיות מצב שבו נשנה את הגרף בזמן שטרד אחר עובד עליו (מחשב את האלגוריתם)

שלב 9 - מימוש שלב 7 בעזרת תבנית (שלב 8) - עשר נקודות

ממש את שלב 7 בעזרת הספריה שמימשת בשלב 8.

שלב 10 - producer consumer - עשר נקודות

הוסף לשרת (שלב 9) טרד אחרון שממתין למצב שבו שטח הCH הוא לפחות 100 יחידות מרובעות.
אם לאחר חישוב CH (כזה שיזם משתמש, אין צורך לבצע חישוב בכל הכנסת קשת בודדת) הגענו למצב כזה יש להעיר את הthread שיוציא הדפסה לstdout בצד השרת.

At Least 100 units belongs to CH

אם לאחר שכבר הגענו למצב שיש CH בגודל 100 יחידות לפחות הCH עכשיו קטן יותר (למשל כתוצאה מהפחתת נקודות בגרף)

At Least 100 units no longer belongs to CH

יש להעיר את הטרד הנוסף בעזרת POSIX cond.