שאלה 1

1.א

נוסיף Signal Handler אשר יתפוס את הסינגל SIGALARM בכל שניה. נחזיק מונה שיספור עד 7, כאשר הגיע ל7 נשלח את התהליך הנוכחי לישון. הזמן של המערכת FCFS יוציא אותו מפעולה ויביא את התהליך הבא שמובטח שירוץ 7 שניות, התהליך שהלך לישון באותו הרגע יחזור לסוף התור וכך נממש את הזמן RR.

הSignal Handler

void SigAlarmHndl(int signum){  
 if(++count == 7){  
 count = 0;  
 sleep(1);  
 }  
}

כאשר Count משתנה גלובלי שמאותחל ל0.

כמו כן בפונקציית המיין נוסיף את השורה:

signal(SIGALARM,SigAlarmHndl);

בתחילת התוכנית.

1.ב

נוסיף משתנה גלובלי, אשר ישמור את מספר החוט הנוכחי והחוט הבא אשר צריך לעבוד. לפי הנתון כל שניה נקבל סיגנל, לכן נתפוס וננהל את החוטים על ידי signal handler אשר תופס את SIGALARM. הsignal handler:

int current\_thread = 0;  
  
  
void SigAlarmHndl(int signum){  
 int reg = current\_thread;  
 if(++current\_thread == 3) current\_thread = 0;  
 if(setJmp(b[reg] == 0)){  
 longJmp(b[current\_thread],1);  
 }  
}

כל פעם נכניס למשתנה לוקאלי reg את החוט אשר אנו נמצאים בו עכשיו (ואם חוזרים אליו אז הערך שsetJump יחזיר יהיה 0. נקדם את המונה של החוט ונשמור אותו בין 0 ל2. לאחר מכן נבדוק אם זו פעם ראשונה שהחוט רץ setJmp יחזיר 0 ונשלח לlongJmp את החוט המקודם.אחרי סיבוב כאשר נשלח לlongJmp את החחוט הראשון שוב, הערך המוחזר לא יהיה 0 הוא יהיה אחד והחוט הראשון ירוץ עוד שניה עד הsignal הבא. מכיוון שאנו רוצים שתהליך זה יתפוס את הסיגנל נפריד להתליכים שונים, ובפקונציית הmain נוסיף שורה בה אנו מכניסים את הhandler שלנו ונוסיף fork כדי שהsignal יתפס על ידי החוטים בלבד.

שאלה 2.

2.א –

לא נכון- חוטים משתפים את מרחב הזיכרון ביניהם. לשניהם גישה לאותה המחסנית לכן כאשר ניגש לאותו משתנה משני חוטים שונים נצטרך להגן עליו באמצעות נעילה.

2.ב –

ייתכן שיהיה deadlock, אם כל חוט אחרי השורה הראשונה יעשה context switch נקבל ששלושת המנעולים נעולים וכל התהליכים ילכו לישון אחרי השורה הבאה.

שינוי לקוד אשר יתקן:

החוטים צריכים לנעול באותו הסדר את המנעולים ( כאשר זה 3 חוטים ניתן להגדיר שינעלו בסדר עולה). לדוגמה:

Thread 1:  
1. LOCK(L1);  
2. LOCK(L2);  
3. // critical section  
4. UNLOCK(L2);  
5. UNLOCK(L1);  
  
Thread 2:  
1. LOCK(L1);  
2. LOCK(L3);  
3. // critical section  
4. UNLOCK(L3);  
5. UNLOCK(L1);  
  
Thread 3:  
1. LOCK(L2);  
2. LOCK(L3);  
3. // critical section  
4. UNLOCK(L3);  
5. UNLOCK(L2);

3.

init\_global\_semaphores(){  
 S glass\_s = 1;  
 S orage\_s = 0;  
 S vodka\_s = 0;  
 S umbrella\_s = 0;  
 S vodOrgSync = 0;  
}  
  
  
  
put\_vodka(){  
 down(glass\_s); // only vodka can change glass  
 add(vodka);  
 up(glass\_s); // open lock  
 up(vodOrgSync);// allow orange (not before vodka)  
 serve\_if\_ready();// check if all ingidients are in  
 up(vodka); // twice because if both orange and umbrella down on vodka  
 up(vodka);  
 down(orange); // sleep untill orange and umbrella are in the glass  
 down(unbrella);  
}  
  
put\_oragne(){  
 down(vodOrgSync); // if vodka is already in  
 down(glass\_s); // lock and add orange  
 add(orange);  
 up(glass\_s); // free lock for other ing  
 serve\_if\_ready();// check if all ingidients are in  
 up(orange); // up twice for 2 function  
 up(orange);  
 down(umbrella); // dont cont untill all ing are in  
 down(vodka);  
}  
  
put\_umbrella(){  
 down(glass\_s);  
 add(umbrella);  
 up(glass\_s);  
 serve\_if\_ready();// check if all ingidients are in  
 up(umbrell);  
 up(umbrell);  
 down(orange);  
 down(vodka);  
}

4.