מערכות הפעלה- מטלה 2

היבה אבו-כף: 323980441

גיל פסי: 206500936

א. ע"י שימוש ב semaphores, כתבו אלגוריתם עבור תהליך/תהליכון הנקרא Cow המעביר את הפרה בכל המתחמים פעם אחת בדיוק תחת הכללים שהוזכרו. השתמשו במתודות הבאות כדי להודיע על כניסה למתחם מסוים:
drink() – הפרה במתחם השוקת
eat() – הפרה במתחם החציר
walk() – הפרה במתחם הטיול
שים לב שהאלגוריתם שנכתב עבור Cow זהה עבור כל N התהליכים של הפרות ברפת.

א.

אלגוריתם:

-ביצירת כל פרה נשייך אותה ל – cowshed , קבוצת הפרות שממתינות לטיפול.

-כל פרה עוברת את התחנות "חציר" -> "שוקת" -> "טיול" כך מובטח לנו שהסדר נשמר וכל פרה תבצע את המסלול כולו בדיוק פעם אחת.

-כמו כן נוודא באמצעות לולאה שאם פרה מנסה לגשת למתחם הטיול טרם חברותיה סיימו , היא תיכנס למצב המתנה עד אשר פרה אחרת תסיים את תחנת השוקת (אין צורך לבדוק את תחנת החציר כיוון שתחנת השוקת נשארה).

-כל פרה תופסת מקום בתחנה הבא לפי שמשחררת את התחנה הנוכחית. זאת ע"מ להימנע ממצב כמו שמתואר כך :

1. ישנה תוכנית עם 2 פרות.

2. פרה1 מגיעה לתחנת החציר ומסיימת לאכול אבל לא מספיקה להיכנס לתחנת השוקת.

3.פרה2 מגיעה עד לשלב הטיול : כרגע אף פרה לא נמצאת ב – cowshed (פרות לא מטופלות) , אף פרה לא נמצאת ב – Hay Area כיוון שפרה1 סיימה לאכול ואף פרה לא נמצאת ב -Water Area כיוון שפרה2 לא הספיקה להיכנס למתחם. לכאורה מצב תקין אבל ידוע לנו שפרה1 עדיין לא סיימה לשתות.

ב.

בנספחים.

ג. מדוע דרוש לנו האובייקט SharedResources ? למה לא ניתן פשוט ליצור את ה semaphores בתוך המחלקה CowThread?

ג.

עלינו ליצור מחלקה נפרדת לסמפורים מכיוון שהם חייבים להיות משותפים לכל הפרות. במקרה בו נגדיר אותם כתכונה, כל פרה תקבל סמפור משלה ולא תהיה מניעה הדדית - mutual exclusion.

שאלה 2 (20 נק')

נתון הקוד הבא, המבצע שימוש בפונקציה fork :

```
void main(){
        int i=0;
        while(i<5){
                pid = fork();
                if (pid == 0)
                        printf("hello");
                else
                        printf("world");
                i++;
        }
}
```

א. כמה פעמים תודפס המילה "hello" וכמה פעמים תודפס מילה "world"?
ב. ציירו את היררכיה של התהליכים שנוצרו.

א.

למעשה כל תהליך מתפצל לשני תהליכים ואלו יתפצלו לעוד שניים. לכן בסה"כ נקבל:

$$\sum_{i=1}^{5} 2^i = \sum_{i=0}^{5} 2^i - 1$$

נשתמש בזהות קומבינטורית:

$$\sum_{i=0}^{n} 2^i = 2^n - 1$$

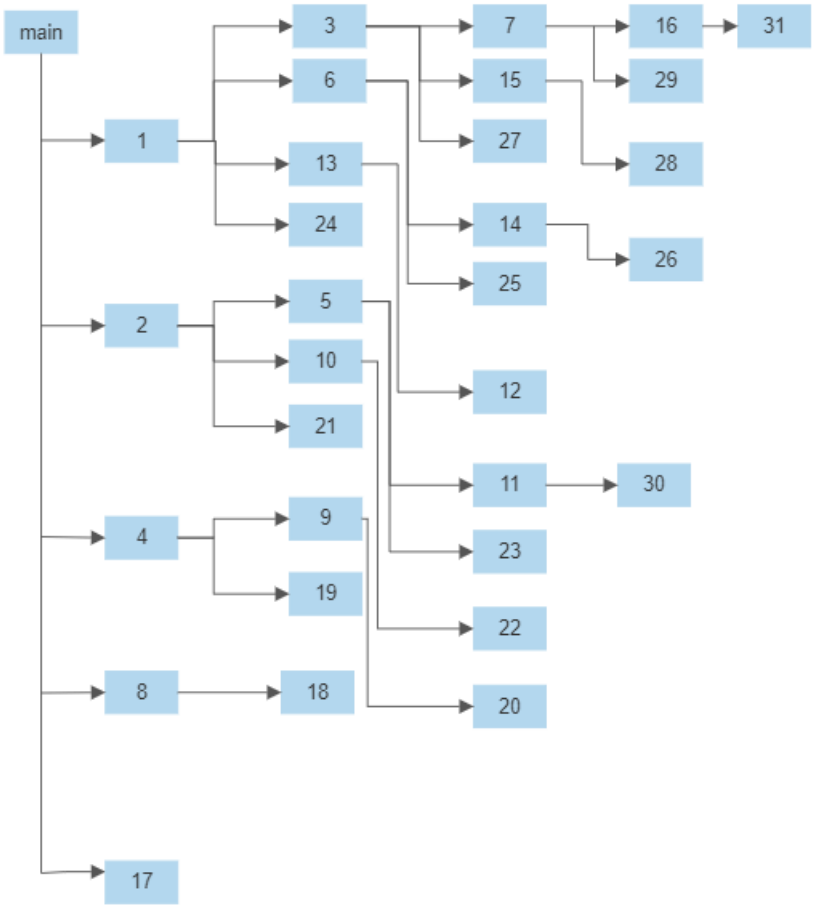
$$\sum_{i=0}^{5} 2^i - 1 = 2^5 - 1 - 1 = 64 - 1 - 1 = 62$$

מאחר ובכל איטרציה קיים תהליך אבא אחד ותהליך בן אחד הפלטים "hello" , "world" , יתחלקו שווה בשווה

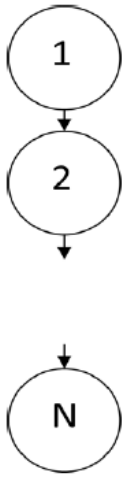
כלומר בסה"כ 31 הדפסות hello ו – 31 הדפסות world .

ב.

עץ היררכיה:


main
1
2
4
8
17
3
6
13
24
5
10
21
9
19
18
7
15
27
14
25
12
11
23
22
20
16
29
28
26
31
30

ג. כתובו תוכנית (בקובץ pdf) אשר תקבל ממשתמש מספר שלם חיובי N ותיצור שרוך תהליכים:

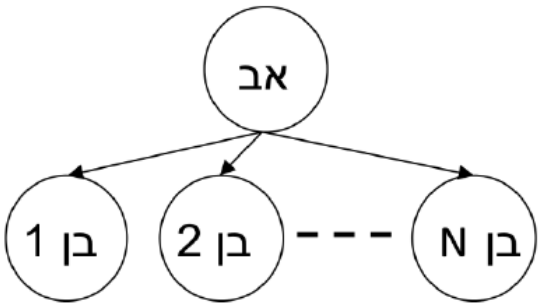


יש להדפיס pid של כל אחד מהתהליכים שנוצרו.

```
#include<iostream>
#include <unistd.h>

void processLace (int n){
    int i=0;
   while(i<n){
   int pid = fork();
   if (pid != 0){
    cout << "Process " <<  getpid() << " died" <<endl ;
    break;
   }
   i++;
  }
}
int main(){
  processLace(8);
  return 0;
}
```

ד. כתוב תוכנית אשר תקבל מהמשתמש מספר שלם חיובי N ותיצור עץ תהליכים בגובה 1 (תהליך אב ו-N בנים שלו):



יש להדפיס pid של כל אחד מהתהליכים שנוצרו.

```
#include<iostream>
#include <unistd.h>


void processTree (int n){
    int i=0;
   while(i<n){
   int pid = fork();
   if (pid = = 0){
    cout << "Process " <<  getpid() << " died" <<endl ;
    break;
   }
   i++;
  }


}
int main(){
  processTree(8);
  return 0;
}
```

שאלה 3 (10 נק')

נתון קטע הקוד:

```
int main(int argc, char *argv[]){

  for(int i = 0; i < 7; i++)   {
        int pid = fork();
        printf("%d : %d , \n", getpid(), i);
        if(pid > 0)
                break;
  }
  return 0;
}
```

הניחו כי ה-pid של התהליך שמתחיל להריץ את ה-main הוא 1, וכי pids ניתנים באופן סדרתי, ואין עוד תהליכים במערכת. כמו כן, הניחו שיטת התזמון First Come First Served. כלומר, תהליך שנכנס לתור ראשון ירוץ ראשון.

א. מה ה-pid הגבוה ביותר שהוקצה במהלך הקוד להלן?
ב. מה ידפיס הקוד (משמאל לימין)?

א.

בסך הכל נוצרים 7 תהליכים . כלומר מספר התהליכון הגבוה ביותר יהיה 1 + 7 = 8 .

ב.

כל תהליך שנכנס ללולאה קודם כל מדפיס את עצמו. במידה והוא שרד, תודפס הודעה נוספת שלו אך בכל מקרה לאחר איטרציה יסיים את פעולתו. לכן הפלט יהיה:

```
1:0,
2:0,
2:1,
3:1,
3:2,
4:2,
4:3,
5:3,
5:4,
6:4,
6:5,
7:5,
7:6,
8:6,
```

שאלה 4 (5 נק')

להלן תבנית של תוכנית עם שני תהליכים ופונקציית main.

main()	p1()	p2()

	{start();
	while (TRUE){
	server_create();
	finish();}
	}
		{ while (TRUE)
		{before();
		client_use();
		cleanup();}}

התניות:

א. קטע start צריך לסיים לרוץ לפני שקטע before ירוץ אפילו פעם אחת.

ב. קטע server_create הוא יצרן וקטע client_use הוא הצרכן של מה שהיצרן יצר.

ג. קטעים before, finish, cleanup הם קטעים קריטיים אחד לשני ואסור שירוצו יחד.

הוסף לפסאודו-קוד הנ"ל פקודות של סמפור כדי לקיים את שלוש ההתניות. השתמש אך ורק בפקודות הבאות: wait(s), signal(s), init_sem(s,n).

Main()	P1()	P2()

Semaphore s1;
Semaphore s2;
 	
init_sem(s1 , 0);
init_sem(s2 , 1);
	{start();
	while(TRUE){
	signal(s1);
	server_create();
	wait(s2);
	finish();
	signal(s2);
	}
	}
		{while(TRUE){
		wait(s1)
		wait(s2);
		before();
		signal(s2);
		client_use();
		wait(s2);
		cleanup();
		signal(s2);
		
		}
		}

הסבר:

הסמפור S1 מייצג את כמות המוצרים שיצר היצרן. כך מימשנו את דרישה ב' . לכן בתחילת הריצה מאותחל ל – 0 כיוון שעדיין לא נוצרו מוצרים , בדרך זו נבטיח הצרכן לא יכול לצרוך לפני שהיצרן מייצר – דרישה א' .

הסמפור S2 דווקא לא מייצג משאבים אלא נעילה עבור הפעולות before,finish,cleanup , לכן ננעל אותו טרם הכניסה לפעולות אלו ולאחר היציאה מהן – דרישה ג' .

שאלה 5 (5 נק')

נתונים 3 תהליכים שרצים במקביל:

main	P1	P2	P3
	{	{	{
	B	A	D
	E	C	F
	}	}	}

התניות:

- קטע A חייב להסתיים לפני שמתחילים B, C ו-D.
- קטעים B, C, D הם קטעי קוד קריטי ואסור שיפעלו באותו זמן.
- קטע E תתבצע רק אחרי סיום B, C ו-D.

תעתיקו את התהליכים לקובץ בתוספת מנגנון שמקיים את ההתניות בעזרת סמפורים.

Main	P1	P2	P3
Semaphore s1;	{	{	{
Semaphore s2;	wait(s1);	A	wait(s1);
	wait(s2);	signal(s1);	wait(s2);
init_sem(s1 , 0)	B	signal(s1);	D
init_sem(s2 , 1)	signal(s2);	wait(s2);	signal(s2);
	wait(s1);	C	signal(s1);
	wait(s1);	signal(s2);	F
	E	signal(s1);	}
	}	}	

הסבר:

הסמפור s1 אחראי לוודא את הדרישה הראשונה. אחרי ביצוע קטע A , נוסיף 2 permissions. אחד בעבור קטע B והשני עבור קטע D . מאחר וקטע C מתבצע בכל מקרה אחרי קטע A , אין צורך לקצות לו אישור מיוחד.

הסמפור s2 מהוו מנעול עבור שלושת הקטעים D,C,B ולכן דורש רק permission אחד .

נשתמש שוב בסמפור s1 על מנת לוודא את דרישה 3, כאשר C או D מסיימים , הם מוסיפים permission ל s1. גם כאן אין צורך לוודא אישור מקטע B בכל מקרה הוא מופיע בקוד לפני קטע E .