מכללת אורט בראודה, המחלקה להנדסת תוכנה

 מערכות זמן אמת - 61133 סמסטר אביב, תשע"ד

בחינה סופית – מועד א'

תאריך 18.7.2016

סגל הקורס: איתן רון, מיכל מהלר, דניאל בואנוס

מותר כל חומר עזר כתוב ומודפס .

זמן הבחינה 3 שעות.

שאלות 1,3 הן שאלות תכנות. שאלה 2 היא שאלה תאורטית.

יש להחזיר את השאלון. אין להעתיק אותו.

טיוטא יש לרשום רק במחברות בחינה, לסמן "טיוטא" ולהגיש עם הבחינה.

במידת הצורך, יש לבצע שימור אוגרים לפי המוסכמות של C.

תן הסבר קצר ותמציתי לכל פתרון. הדבר יקל על הבדיקה, בייחוד במקרים של פתרונות לא לגמרי נכונים.

**שאלה מספר 1 (34 נקודות)**

עליך לכתוב תוכנית DOS העושה את הדברים הבאים:

* המסך יהיה ריק ושחור
* מציבה את ה-cursor בקצה השמאלי העליון
* מגדילה את ה-cursor למקסימום (מכסה את כל המשבצת)
* דואגת שה-cursor יהיה **ירוק מודגש** (זכור שצבע ה-cursor הוא צבע האות במשבצת).
* לחיצה על החיצים "חץ למעלה/למטה/ימינה/שמאלה" מזיזה את ה-cursor בהתאם כאשר הוא משאיר שובל **לבן** על המשבצות שהו היה בהם.
* כל 20 שניות המסך יחזור להיות שחור.

***פתרון:***

*צריך לממש פוינטר, נקרא לו b800h שמצביע לסגמנט 0b800h היסט 0, להציב 0xA לכל בית שני.*

*בקובץ* [*http://math.haifa.ac.il/ronn/realtime/docs/part1.pdf*](http://math.haifa.ac.il/ronn/realtime/docs/part1.pdf) *ניתן למצוא:*

*פורמט המאפיינים בעמוד 16*

*את הפקודות למיקום ה-cursor בעמוד 18*

*מזיזים את ה-cursor בהתאם למקשי החיצים, כאשר כל משבצת ה-cursor עוזב מציבים למאפיינים את הערך 0x7A*

*בפסיקת השעון כל פסיקה 364 מאתחלים מחדש את המסך*

**שאלה מספר 2 (33 נקודות)**

נניח שמערכת Hard Real Time (להלן HRT) עובדת עם מטריצה גדולה n x m שידוע שרוב איבריה אפס. קיים מבנה נתונים ואלגוריתמים המאפשרים ייצוג **דליל** של המטריצה כלומר ייצגו ויפעלו רק על האיברים **שוני האפס** של המטריצה. למשל אפשר לממש מערך דינמי של רשומות {i,j,Vij} ממוינים קודם לפי i ואחר כך לפי j ולחשב מטריצת סכום באלגוריתם דומה למיזוג.

1. באיזה תנאים ניתן פשוט להתעלם מהאופי הדליל של המטריצה ולממש את מבני הנתונים המערכיים הדו-מימדיים הרגילים והאלגוריתמים הרגילים? נמק.
2. באיזה תנאים לדעתך כדאי יהיה לעשות זאת? נמק.
3. במידה וחייבים להסתמך על האופי הדליל של המטריצות, באיזה תנאים פתרון כזה יכול להיחשב HRT? נמק.
4. תאר סכמה של בדיקה המאפשרת לבדוק מערכת בתנאים של 3 בצורה כזו שיהיה לבדוק את המערכת למשך חודש יהיה אפשר להכריז עליה כמערכת העומדת בתנאים המצופים של HRT.

***פתרון:***

*הרעיון הבסיסי די פשוט:*

*1.      אם הטיפול במטריצה היא חלק מפעולה מחזורית וקוד נאיבי עומד בלוח הזמנים אין שום צורך ליעל אותו.*

*2.      אם העמידה בזמנים הוא עם עודף זמן עדיף לממש משהו פשוט ודטרמיניסטי.*

*3.      אם האלגוריתמים הנאיביים לא עומדים בלוח הזמנים, חייבים להסתמך על הדלילות של המטריצה, בתנאי שיש חסם ודאי (דגש ודאי) על מספר האיברים שוני האפס של המטריצה.  זה קורה במספר לא מבוטל של מקרים, למשל אם המטריצה משקפת קשרים בין מחלקות של רשות, יתכן שכל מחלקה יש רק מספר קבוע של מחלקות אחרות שהוא קשור אליהם. תשובה של חלק מהסטודנטים שהתקבלה היא אם התוכן של המטריצה קבוע. זה אכן מקרה פרטי חשוב.*

*4.      בזמן בדיקות המערכת, כדאי לממש קוד המבזבז זמן בהתאם לחסם הוודאי ולא לפי הקלטים הנבדקים. בצורה כזו אפשר למנוע מצב שבו ייתכן שמקרה גרוע נדיר לא נבדק. כאילו להגדיל את המרכיב שונה האפס באופן מלאכותי. במילים אחרות: לוודא הרצה על מקרים גרועים בלבד.*

**שאלה מספר 3 (33 נקודות)**

בשאלה הזו עליך להשתמש במאגר זיכרון מהסוג שנלמד בהרצאה ובמעבדה 7 על מנת לממש מושג של Cached Function.

מצד אחד Cached Function מייצגת פונקציה f ומצד שני יש לה מבנה נתונים השומר בתוכו זוגות של ערכים {(x,f(x)} כך שכאשר המושג נדרש לחשב את f(x) הוא בודק אם x הוא אחד מהערכים שהוא התבקש לחשב. במידה וכן הוא קורא את הערך המחושב ממבנה הנתונים ומבנה הנתונים אינו משתנה כלל. במידה והערך אינו שם הוא מחשב אותו ומוסיף אותו למבנה הנתונים.

הזוגות {(x,f(x)} יישמרו ברשימה מקושרת של חוצצים המתקבלים ממאגר זיכרון. כל חוצץ יכיל מקום ל-pair\_no זוגות של מספרי double + מצביע לחוצץ הבא. בכל פעם שיש צורך בחוצץ נוסף מבקשים אותו מהמאגר ומקשרים לרשימה הקיימת.

כאשר יש בקשה לחשב ערך של הפונקציה, סורקים את החוצצים לראות אם הערך חושב כבר, מספר הזוגות הפעילות בחוצץ האחרון הוא (מספר הזוגות בכל חוצץ) % (מספר הזוגות הפעילים בחוץ) למעט כאשר המספר הזה הוא 0, אז הזוגות הפעילות בחוצץ האחרון הוא מספר הזוגות בכל חוצץ.

לדוגמא, ה-Cached Function הנוצר על ידי תוכנות הדוגמא שתהיה לעיל נראית כך:



מימוש Cached Function יהיה מבוסס על הרשימה הבאה:

typedef struct cached\_function\_rec

typedef double (\*FPTR)(double);

typedef struct cached\_function\_rec

{

 FPTR fp;

 int pair\_no;

 int buff\_size;

 void \*first;

 int pool\_id;

 int num\_of\_active\_buffs;

 int max\_num\_of\_active\_buffs;

 int no\_of\_active\_pairs;

} CACHED\_FUNCTION\_REC, \*CACHED\_FUNCTION\_REC\_PTR;

עליך לכתוב רוטינה:

void init\_cached\_function\_rec(CACHED\_FUNCTION\_REC\_PTR rptr,

 FPTR fp, int pair\_no,

 int number\_of\_buffs);

המאתחלת את הרשומה המוצבעת ע"י rptr כולל בקשה להקצאת מאגר מתאים.

עליך לכתוב רוטינה:

double compute\_value(CACHED\_FUNCTION\_REC\_PTR rptr,

 double x)

המחזירה את fp(x) אם ע"י שליפה מהמאגר ואם ע"י חישוב והוספה למאגר.

לדוגמא, הפלט של התוכנית הבאה:

double sqr(double x)

 {

 return x\*x;

 } // sqr

int main()

{

 int i;

 poolinit();

 CACHED\_FUNCTION\_REC cfr1;

 double x, d;

 init\_cached\_function\_rec(&cfr1, sqr, 3, 5);

 for(i=1; i< 7; i++)

 {

 x = 1.01\*i;

 d = compute\_value(&cfr1, x);

 printf(" sqr(%8.5lf) = %8.5lf\n", x ,d);

 } // for

 printf("\n");

 for(i=3; i< 14; i++)

 {

 x = 1.01\*i;

 d = compute\_value(&cfr1, x);

 printf(" sqr(%8.5lf) = %8.5lf\n", x ,d);

 } // for

 printf("\nCached function sqr:\n");

 print\_cfr(&cfr1);

return 0;

} // main

תהיה:

sqr( 1.01000) = 1.02010

 sqr( 2.02000) = 4.08040

 sqr( 3.03000) = 9.18090

 sqr( 4.04000) = 16.32160

 sqr( 5.05000) = 25.50250

 sqr( 6.06000) = 36.72360

 sqr( 3.03000) = 9.18090

 sqr( 4.04000) = 16.32160

 sqr( 5.05000) = 25.50250

 sqr( 6.06000) = 36.72360

 sqr( 7.07000) = 49.98490

 sqr( 8.08000) = 65.28640

 sqr( 9.09000) = 82.62810

 sqr(10.10000) = 102.01000

 sqr(11.11000) = 123.43210

 sqr(12.12000) = 146.89440

 sqr(13.13000) = 172.39690

Cached function sqr:

 ( 1.01000, 1.02010) ( 2.02000, 4.08040) ( 3.03000, 9.18090)

 ( 4.04000, 16.32160) ( 5.05000, 25.50250) ( 6.06000, 36.72360)

 ( 7.07000, 49.98490) ( 8.08000, 65.28640) ( 9.09000, 82.62810)

 (10.10000, 102.01000) (11.11000, 123.43210) (12.12000, 146.89440)

 (13.13000, 172.39690)

***פתרון***

*החלק החכם פה הוא לקרוא ל-mkpool:*

*poolid =mkpool(n\*sizeof(double)+sizeof(Void \*), no\_of\_buffs);*

*מעבר לזה זה פשוט לממש רשימה מקושרת ע"י קריאות ל-getbuf, עם ה-poolid ש-mkpool החזיר. צריך לעשות אריטמטיקה של פוינטרים (או להלביש רשומה) על השטח שהוקצאץ*