מכללת אורט בראודה, המחלקה להנדסת תוכנה

מערכות זמן אמת - 61133 סמסטר אביב, תשע"ד

בחינה סופית – מועד א'

תאריך 14.2.2015

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)**

**שאלה מספר 2 (33 נקודות)**

על מערכת המחשוב של מכונית מוטלות המשימות הבאות:

1. מנגנון המודד את ההתנגדות לגלגלים של המכונית ובהתאם לזאת מחשבת ומיישמת את הריכוז האופטימלי של אדי הדלק (ריכוז גבוהה עוצמה גדולה יותר במנוע).
2. מנגנון המזהיר את הנהג בפני התנגשות אפשרית של המכונית.
3. מנגנון הבודקת מצב המנוע ובמקרה של התחממות יתר – גורם לכיבוי חרום של המנוע.
4. תאר כיצד היית מחלק את המשימות הללו **בין 2 מחשבים**. הסבר מדוע בהקשר הזה כדאי שהעבודה תתחלק בין 2 מחשבים נפרדים.
5. הסבר למה השיקולים היו אחרים אם, במקום כבוי חירום של המנוע המערכת הייתה צריכה, במקרה של התחממות המנוע, רק להזהיר את הנהג ולא לכבות על דעת עצמה את המנוע. איזה תכונות היו במקרה שהמערכת בסעיף ב' הייתה ממומשת במחשב אחד?

**שאלה מספר 3 (33 נקודות)**

בשאלה זו עליך להשתמש ברוטינות של מאגרי זיכרון על מנת לייצר רשימה מקושרת לינארית ממוינת.

החוצצים המוקצים ע"י getbuf חייבים להיות בדיוק בגודל הנחוץ לייצג את הצמתים ברשימה המקושרת.

הקורא לרוטינות שלך חייב לספק את גודל שדה המידע, המספר המרבי של צמתים ברשימה המקושרת וכן פוינטר לפונקציה המשווה 2 שדות מידע ומחזירה 1 אם הראשון גדול ממש מהשני.

לרשותך עומד הקוד הבא:

typedef int (\*FPTR)(void \*,void \*);

typedef struct slist\_rec

{

int poolid;

int buffsize;

int data\_size;

FPTR fptr;

void \*first;

} SLIST\_REC, \*SLIST\_REC\_PTR;

רשומה מסוג SLIST\_REC תייצג את הרשימה המקושרת.

השדות poolid יכיל את מספר המאגר, buffsize את גודל החוצץ בתוך המאגר, data\_size את גודל שדה המידע, fptr את הפוינטר לפונקציה ו-first יצביע על הצומת הראשון ברשימה.

עליך לממש את הרוטינות הבאות:

SLIST\_REC slist\_init(int data\_size, int max\_node\_num, FPTR fptr);

המאתחלת את המאגר ומחזירה את הרשומה המייצגת אותה, כאשר max\_node\_num הוא המספר המרבי של צמתים ברשימה, data\_size את גודל שדה המידע, fptr את הפוינטר לפונקציה.

void insert\_slist(char \*info, SLIST\_REC\_PTR slist\_rec);

המקצה חוצץ ומכניסה את המידע המוצבע ע"י info לתוך מאגר המייצג ע"י הרשומה המוצבעת ע"י slist\_rec.

void extract\_info(char \*dest, char \*src, SLIST\_REC\_PTR slist );

המחלצת את המידע מתוך חוצץ המוצבע ע"י src לתוך שטח המוצבע ע"י dest בתוך רשימה תוך מאגר המייצג ע"י הרשומה המוצבעת ע"י slist\_rec.

char \*next\_ptr(char \*src, SLIST\_REC\_PTR slist);

רוטינה המחלצת את הפוינטר לעוקב מתוך צומת המוצבע ע"י src

לדוגמא, הפלט של התוכנית הבאה:

int compare\_double(void \*ptr1, void \*ptr2)

{

double d1, d2;

memcpy(&d1, ptr1, sizeof(double));

memcpy(&d2, ptr2, sizeof(double));

if(d1 > d2)

return 1;

else

return 0;

} // compare\_double

int compare\_long(void \*ptr1, void \*ptr2)

{

long int l1, l2;

memcpy(&l1, ptr1, sizeof(long int));

memcpy(&l2, ptr2, sizeof(long int));

if(l1 > l2)

return 1;

else

return 0;

} // compare\_long

int main()

{

double d;

long int l;

int i;

char \*node= NULL;

SLIST\_REC slist;

SLIST\_REC slist2;

poolinit();

printf("\nslist:\n");

slist = slist\_init(sizeof(double), 10, compare\_double);

for(i=1; i <=13; i +=3)

{

d = i\*1.1;

if ((i % 2) == 0)

d = -d;

insert\_slist((char \*)&d, &slist);

} // for

for(i=2; i <=14; i +=3)

{

d = i\*1.1;

if ((i % 2) == 0)

d = -d;

insert\_slist((char \*)&d, &slist);

} // for

node = slist.first;

for(i=1; i <=10; i++)

{

extract\_info((char \*)&d, (char \*)node, &slist);

node = next\_ptr((char \*)node, &slist);

printf(" %8.3lf ", d);

} // for

printf("\n");

printf("\nslist2:\n");

slist2 = slist\_init(sizeof(long int), 10, compare\_long);

for(i=1; i <=13; i +=3)

{

l = i;

if ((i % 2) == 0)

l = -l;

insert\_slist((char \*)&l, &slist2);

} // for

for(i=2; i <=14; i +=3)

{

l = i;

if ((i % 2) == 0)

l = -l;

insert\_slist((char \*)&l, &slist2);

} // for

node = slist2.first;

for(i=1; i <=10; i++)

{

extract\_info((char \*)&l, (char \*)node, &slist2);

node = next\_ptr((char \*)node, &slist2);

printf(" %8ld ", l);

} // for

printf("\n");

return 0;

} // main

יהיה:

slist:

-15.400 -11.000 -8.800 -4.400 -2.200 1.100 5.500 7.700

12.100 14.300

slist2:

-14 -10 -8 -4 -2 1 5 7

11 13