מבוא למדעי מחשב מ' / ח' (234114 / 234117)

סמסטר אביב תשס"ו

פתרון מבחן מסכם מועד ב', 29 ספטמבר 2006

מספר סטודנט	*שם משפחה*	*שם פרטי*

משך המבחן: 3 שעות.
חומר עזר: אין להשתמש בכל חומר עזר בכתב, מודפס או אלקטרוני.

הנחיות והוראות:

- מלאו את הפרטים בראש דף זה.
- בדקו שיש 16 עמודים (5 שאלות) במבחן, כולל עמוד זה.
- כתבו את התשובות על טופס המבחן בלבד, במקומות המיועדים לכך. שימו לב שהמקום המיועד לתשובה אינו מעיד בהכרח על אורך התשובה הנכונה.
- העמודים הזוגיים בבחינה ריקים. ניתן להשתמש בהם כדפי טיוטה וכן לכתיבת תשובותיכם. סמנו טיוטות באופן ברור על מנת שהן לא תיבדקנה.
- יש לכתוב באופן ברור, נקי ומסודר
- אין לכתוב הערות והסברים לתשובות אם לא נתבקשתם מפורשות לכך.
- בכל השאלות, הינכם רשאים להגדיר (ולממש) פונקציות עזר כרצונכם.
- **אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה אלא אם צוין אחרת בשאלה.**
- **פתרון שלא עומד בדרישות הסיבוכיות יקבל ניקוד חלקי בלבד.**

שאלה	ערך	הישג	בודק
1	20		
2	20		
3	20		
4	20		
5	20		
סה"כ	100		

צוות הקורסים 234114/7
מרצים: סאהר אסמיר, פרופ' חיים גוטסמן (מרצה אחראי).
מתרגלים:, אייל רוזנברג, מירי בן חן, אולג רוכלנקו, רן רובינשטיין, שיאון שחורי, שי אוחיון (מתרגל אחראי). **בודקי תרגילים**: מרק גינזבורג, מאשה ניקולסקי

בהצלחה!

שאלה 1 (20 נקודות)

סטודנטים בקורס "מבוא למדעי מחשב למתקדמים" בטכניון בנו מחשב חדיש אך שכחו לתמוך במספרים מטיפוסים float או double. כדי לעזור להם עליכם לכתוב פונקציה ללא שימוש בטיפוסים הנ"ל אשר מקבלת שבר חיובי בין 0 ל-1 בצורה של מונה (a) ומכנה (b), ואת מספר הספרות (d) להדפסה אחרי הנקודה העשרונית. על הפונקציה להדפיס d אלו.

שימו לב כי אסור להשתמש במספרים ממשיים וגם אין צורך להחזיר את המספר, אלא רק להדפיסו. לדוגמה, עבור הקלט a=2, b=5, d=4 הפונקציה צריכה להדפיס "0.4000". שימו לב כי **אין צורך** לעגל את התוצאה. לדוגמא עבור a=75,999 ו b=100,000 ו d=4 יש להדפיס "0.7599"

הערה כללית: אין להוסיף טיפול מיוחד בגלישות של ערכים מחוץ לתחום הייצוג של הטיפוסים.

א. ממשו את הפונקציה בסיבוכיות זמן O(d) ובסיבוכיות מקום (נוסף) O(d).
רמז: חילוק ארוך.

```
void print_double(int a, int b, int d){

  int dig = 0, dignum;

  if (a==b) {
    printf("1.");
    a = 0;
  }
  else {
    printf("0.");
  }

  for (dignum=0; dignum < d; dignum++) {
    a *= 10;
    dig = a / b;
    printf("%d",dig);
    a = a - dig * b;
  }
}
```

שאלה 2 (20 נקודות)

הערה: כל המספרים המופיעים בשאלה הם מספרים שלמים, ובכל המערכים האיברים שונים זה מזה.
נאמר שמערך הוא סימטרי סביב k, אם לכל מספר x שבמערך גם הערכים $\{k + |x - k|, k - |x - k|\}$ הם ערכים במערך.
במילים אחרות, אם נצייר את כל האיברים במערך על גבי ציר, נקבל תמונה סימטרית סביב k.
לדוגמה, המערך

1	11	5	13	7	9	3	15

הוא סימטרי סביב 8.
שימו לב: מערך סימטרי סביב k לא בהכרח ממוין.

נתון מערך A באורך n. ידוע כי קיים k שלם עבורו המערך A סימטרי סביב k. עליכם לכתוב פונקציה בשם FindSymmetricK, אשר מקבלת את המערך, ואת אורכו, ומחזירה את k, בסיבוכיות זמן O(n) ובסיבוכיות מקום (נוסף) O(1).

```
int FindSymmetricK(int *A, int n)
{
  int min, max, i;
  min = max = A[0];
  for (i = 1; i < n; i++) {
    min = (A[i] < min) ? A[i] : min;
    max = (A[i] > max) ? A[i] : max;
  }
  return (min+max) / 2;
}
```

ב. בהמשך לסעיף הקודם, נגדיר מערך **סופר-סימטרי** באופן רקורסיבי.

- כל מערך באורך 1 הוא סופר-סימטרי.
- מערך A באורך n>1 הוא סופר-סימטרי, אם קיים k עבורו A סימטרי סביב k, וכן כל איברי A הקטנים מ k הם תת מערך סופר-סימטרי וכן כל אברי A הגדולים מ k הם תת מערך סופר סימטרי.

לדוגמא, המערך מסעיף א' הינו סופר-סימטרי, כי עבור k=8 נקבל תת מערך {1,5,3,7} שהוא סופר סימטרי עבור k=4, ותת מערך נוסף {11,13,9,15} שהוא גם סופר סימטרי סביב k=12.

הערה: המערכים בשאלה הם באורך שהוא חזקה של 2.
עליכם לכתוב פונקציה **<u>רקורסיבית</u>** בשם SortSuperSymmetric אשר מקבלת מערך סופר-סימטרי A באורך n, וממיינת אותו בסיבוכיות זמן O(n) וסיבוכיות מקום O(log n).

לצורך המימוש ניתן להשתמש בפונקציה pivot (שחתימתה נתונה בהמשך), אשר מקבלת מערך A, את אורכו n, ומספר x, ומסדרת את המערך כך שכל האיברים שקטנים מ- x יימצאו לפני כל האיברים שגדולים מ- x. סיבוכיות הזמן של הפונקציה pivot היא O(n), וסיבוכיות המקום O(1).
כמו כן, ניתן להשתמש בפונקציה מסעיף א', גם אם לא מימשתם אותה.

```
void pivot(int *A, int n, int x);
```

```
void SortSuperSymmetric(int *A, int n)
{
  int i, k;
  if (n <= 1) {
    return;
  }
  k = FindSymmetricK(A, n);
  pivot(A, n, k);
  SortSuperSymmetric(A, n/2);
  for (i=0; i < n/2; i++)
    A[n-i-1] = A[i] + (k-A[i])*2;
}
```

שאלה 3 (20 נקודות)

נתון מערך A של מספרים שלמים הממויין בסדר עולה. אלגוריתם החיפוש שנעסוק בו בשאלה זו נקרא "חיפוש אינטרפולציה". האלגוריתם דומה לחיפוש בינארי, אלא שבמקום לחפש את המספר במרכז סדרת המספרים A[i] ,.., A[j], דהיינו במקום k (כך ש- $k=(i+j)/2$), בכל שלב מחפשים k המקיים:

$$\frac{b-A[i]}{A[j]-A[i]} = \frac{k-i}{j-i}$$

כאשר b הוא הערך אותו מחפשים. האלגוריתם מחקה את שיטת החיפוש המוכר בתוך מילון: אם אנו מחפשים את המילה "ביתן", נפתח תחילה את המילון קרוב יותר להתחלה מאשר לסוף, מאחר והאות "ב" קרובה יותר לאות "א" מאשר לאות "ת". בחיפוש בינארי, לעומת זאת, היינו פותחים באמצע המילון.

כתבו פונקציה **רקורסיבית** המבצעת חיפוש אינטרפולציה של הערך b במערך A באורך n. הקפד לרשום תנאי סיום כללי ומדויק. על הפונקציה להחזיר 1 אם הערך b נמצא במערך A או 0 אחרת.

```
int interp_search(int *A, int n, int b)
{
  int mid;
  if (n<=0) return 0;
  if (n==1) return (A[0]==b);

  mid = (b - A[0]) / (double)(A[n-1]-A[0]) * (n-1);
  if (A[mid] == b) return 1;
  if (A[mid] > b) return interp_search(A,mid-1,b);
  return interp_search(A+mid+1, n-(mid+1), b);
}
```

שאלה 4 (20 נקודות)

כתבו פונקציה המקבלת מערך A של מספרים שלמים בגודל n, **ומדפיסה** את הרצף הארוך ביותר של מספרים עוקבים חיוביים ממש המופיעים במערך A. הפונקציה תחזיר את אורך הרצף.
אם ישנם כמה רצפים באורך שווה, ניתן להדפיס כל אחד מהם – אבל, רק אחד מהם.
אם אין אף מספר חיובי במערך A, על הפונקציה להחזיר 0.
על הפונקציה לרוץ בסיבוכיות זמן O(n), ובסיבוכיות מקום (נוסף) O(1).

דוגמאות:

A={1, 5, -4, -3, -2, **6, 7**} :

הפונקציה תדפיס " 7 6" ותחזיר את הערך 2.

A={**1, 12, 3**, **2**, 0, -9, -2, -1, 0, 1, 2}

הפונקציה תדפיס " 2 3 12 1" ותחזיר את הערך 4.
(ניתן להדפיס רווח נוסף בסוף השורה על מנת לקבל מימוש פשוט יותר).

```
int PrintLongestSequence(int *A, int n) {

  int i, maxpos, len=0, maxlen=0;

  for (i=0; i<n; i++) {
    if (A[i]>0) {
      len++;
      if (len > maxlen) {
        maxlen = len;
        maxpos = i - len + 1;
      }
    }
    else len = 0;
  }

  for (i=0; i<maxlen; ++i)
    printf("%d ",A[i+maxpos]);

  return maxlen;
}
```

שאלה 5 (20 נקודות)

"פנגרמה" הוא משפט המכיל את כל האותיות באלף-בית לפחות פעם אחת. לדוגמא, המשפטים הבאים הם פנגרמות באנגלית:

The quick brown fox jumps over the lazy dog
The five boxing wizards jump quickly

והמשפט הבא הוא פנגרמה בעברית:
"דג סקרן שט בים מאוכזב ולפתע מצא חברה"

הניחו שנתון מערך גלובלי words שמכיל את כל המלים באנגלית, ושגודלו הוא הקבוע NUM_WORDS.

כתבו פונקציה המדפיסה את כל הפנגרמות באנגלית הקצרות מ 100 תווים.

הנחיות:
- על הפנגרמה להכיל אותיות קטנות ורווחים בלבד.
- הניחו שקיימת פונקציה int is_sentence(char* str) שמחזירה 1 אם המחרוזת שהיא קיבלה היא התחלה של משפט חוקי באנגלית.
- על הפתרון **להיות רקורסיבי, ולעבוד בשיטת ה backtracking**. אין להמשיך ברקורסיה עם משפט שברור שהוא לא פנגרמה כי הוא אינו משפט חוקי.
- מומלץ לכתוב פונקציית עזר הבודקת אם משפט הוא אכן פנגרמה.

יש להשתמש במערך הגלובלי words, בגודל NUM_WORDS אשר מוגדרים כך:

```
#define NUM_WORDS 1000
char *words [NUM_WORDS] = {…}
```

.

```
void PrintAllPanagrams() {
  char p[101] = {0};
  PrintAllPangramsAux(p, 0);
}
```

```
void PrintAllPangramsAux(char *p, int len)
{
  int i, wordlen;
  p[len] = 0;
  if(!is_sentence(p))
    return;

  if (ispangram(p))
    printf("%s\n", p);

  for (i=0; i<NUM_WORDS; i++)
  {
    wordlen = strlen(words[i]);
    if (len + wordlen + (len>0) > 100)  // if len>0,
      continue;                         // need to add a ' '
    if (len>0)
      p[len++] = ' ';
    strcpy(&p[len], words[i]);
    PrintAllPangramsAux(p, len+wordlen);
    if (len>0)
      len--;
  }
}


int ispangram(char *p)
{
  int letters=0;
  int present[26] = {0};

  for ( ; *p ; p++) {
    if (*p > 'z' || *p < 'a')
      continue;
    if (!present[*p-'a']) {
      present[*p-'a'] = 1;
      letters++;
    }
  }
  return (letters==26);
}
```

```
int strlen(char *s) {
  int len = 0;
  while (s[len])
    len++;
  return len;
}


void strcpy(char *t, char *s) {
  do {
    *t = *s;
    t++; s++;
  } while (*s);
}
```