סמסטר ג' תשס"ד
מועד: א' 23/09/2004
משך הבחינה: 3 שעות
חומר עזר: שני דפי עזר

בחינה בקורס: **תוכנה 1**
פרופ' דן כהן-אור ואפי פוגל

הנחיות כלליות לבחינה:

- המבחן מורכב משלוש שאלות תכנות ושתי שאלות "אמריקאיות". יש להשיב על כל השאלות.
- חובה לתעד כל פעולה לא טריוויאלית שנעשית.
- יש לכתוב קוד קריא ויעיל ככל האפשר.
- נא לכתוב בכתב קריא ולא מחובר.

בהצלחה !

ת.ז.:

טבלת תשובות לחלק האמריקאי

	e	d	c	b	a
שאלה 4				X	
שאלה 5	X				

הערות לתשובות בחלק האמריקאי

אמנם רק תשובה נכונה תזכה בניקוד עבור כל שאלה, אולם ניתן לצרף לכל תשובה אמריקאית הסבר קצר. ההסבר לא ייבדק במסגרת הבדיקה הרגילה, אך ניתן יהיה להסתמך עליו במסגרת ערעור, אם יידרש. מומלץ לצרף הסבר לתשובה אמריקאית במיוחד במקרים קיצוניים בהם **נראה לך** שתשובתך נכונה, אך **נראה לך** שהיא איננה התשובה שאליה התכוון המרצה. מובן שהסבר שגוי או בלתי סביר לא יועיל בכל מקרה (אך גם לא יזיק, אם ממילא סימנת את התשובה הנכונה).

שאלה 4: רק בסעיף השמורת הלא יש הפרה של התוויה הבלתי תלויה
לא תלויה

שאלה 5: מתחושה כללית" הקוד על הרחבות והגבלה לא היה לו
על ה בלתי אלעיל

שאלה מס' 1 (40 נקודות)

כתוב מימוש חלקי לתוכנית המקבלת מחרוזת ארוכה ומדפיסה את מספר המופעים של המילה השכיחה ביותר, ואת מספר המילים השכיחות ביותר. מותר להניח (בלי לבדוק) כי מחרוזת הקלט מסתיימת ב- 0\ ויכולה להכיל אותיות קטנות וגדולות, רווחים, ותווים אחרים כולל n\. מילה היא רצף של אותיות גדולות או קטנות, ללא רווחים, ספרות או תווים אחרים. לצורך השוואה בין מילים, אין להבדיל בין אותיות גדולות לקטנות. לדוגמא, עבור מחרוזת הקלט הבאה:

```
What's in a Name? that which we call a rose
By any other name would smell as sweet;
So Romeo would, were he not Romeo1 call'd,
```

התוכנית מדפיסה:

```
2 5
```

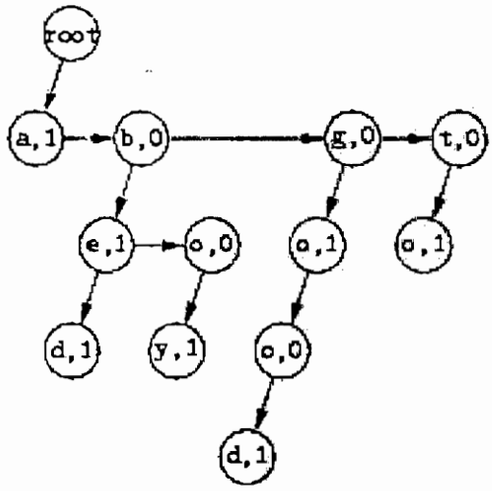
כלומר, יש 5 מילים שונות שכל אחת מהן מופיעה פעמיים במחרוזת והן:

```
a, call, name, romeo, would
```

התוכנית משתמשת בעץ השכיחויות: כל המילים מיוצגות כמסלולים בעץ וכל צומת בעץ מתואר על ידי מבנה הנתונים הבא:

```
typedef struct node {
  struct node * leftmost_child;
  struct node * sibling;
  char c;
  int words;
} Node;
```

שדה c בכל צומת פרט לשורש מכיל אות המופיעה במילה. מסלול מהשורש לצומת כלשהי מתאר מילה. ערך השדה words בצומת כלשהי מכיל את מספר הפעמים שהמילה המסתיימת בצומת זה מופיעה במחרוזת הקלט. שדה leftmost_child מצביע לבן השמאלי ביותר של הצומת, והשדה sibling מצביע לאח הבא מימין. רשימת הצמתים האחים מסודרת בסדר לקסיקוגרפי עולה של האותיות.



העץ המצויר מצד שמאל מיצג את המחרוזת:

"be a good boy, go to bed"

א. (15 נקודות) כתוב פונקציה בעלת המפרט (prototype) הבא:

```
Node * build(char * string);
```

הבונה את עץ השכיחויות. הפונקציה עוברת על כל המילים במחרוזת הקלט, מקצה זכרון כנדרש, ומוסיפה אותם לעץ .

ב. (15 נקודות) כתוב פונקציה בעלת המפרט (prototype) הבא:

```
void frequent_words(Node * root);
```

העוברת על עץ השכיחויות מהשורש ומדפיסה את מספר המופעים של המילה השכיחה ביותר, ואת מספר המילים השכיחות ביותר.

שאלה מס' 2 (15 נקודות)

כתוב פונקציה בעלת המפרט (prototype) הבא:

```
char * strrstr(const char * str, const char * substr);
```

המחזירה מצביע למופע האחרון של המחרוזת substr במחרוזת str. אם המחרוזת substr לא נמצאת, הפונקציה מחזירה NULL.

שאלה מס' 3 (30 נקודות)

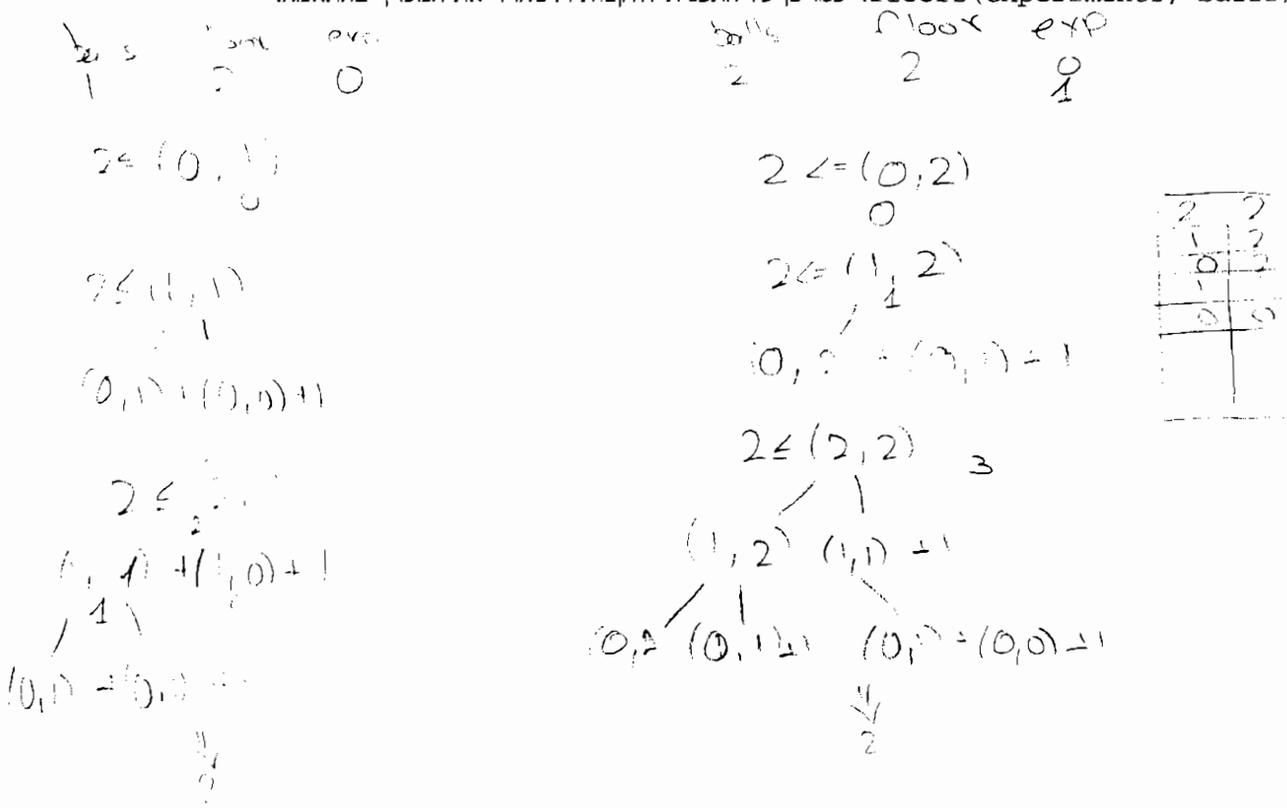
התכנית הבאה מקבלת כקלט מספר קומות in_floors ומספר כדורים in_balls, ומחשבת את מספר הניסוים experiments הקטן ביותר, כך שהתנאי הבא מתקיים:

```
in_floor <= floors(experiments, in_balls)
```

```
#include <stdio.h>

unsigned int floors(unsigned int experiments, unsigned int balls)
{
  if ((balls == 0) || (experiments == 0)) return 0;
  return floors(experiments-1, balls) + floors(experiments-1, balls-1) +1;
}

int main()
{
  unsigned int in_floors, in_balls;
  unsigned int experiments, floor;

  scanf("%d %d", &in_floors, &in_balls);
  for (experiments = 0; 1; experiments++) {
    if (in_floor <= floors(experiments, in_balls)) break;
  }
  printf("experiments = %d\n", experiments);
  return 0;
}
```

אופן החשוב אינו יעיל כלל, מפני שהפונקציה floors(experiments, balls) הינה פונקציה רקורסיבית הקוראת לעצמה פעמיים, והיא נקראת מתוך לולאה.

שכתב את התכנית כך שאותה תוצאה תחושב באופן יעיל. על התכנית להמנע מחישובים עוקבים של floors(experiments, balls) עם אותם פרמטרים, על ידי שימוש במערך זמני אשר יכיל את ערכי floors(experiments, balls). כמו כן על התכנית להקצות ולשחרר את המערך בהתאמה.



שאלה מס' 4 (10 נקודות)

נתון הביטוי הרגולרי הבא:

```
m/\b[a-zA-Z0-9._%-]+@[a-zA-Z0-9._%-]+\.([a-zA-Z0-9._%-]{2,4})\b \1/)
```

והמחרוזות הבאות:

1. blah@foo.com com
2. blah@foo.com
3. ishekhad@makomechad.co.il co.il
4. ab@bc.cd cd
5. a@b.c c

אילו מחרוזות מתאימות לביטוי הרגולרי?

a) 1, 2, 3, 4 ו-5
b) 1, ו-4
c) 2
d) 1, 3, ו-4
e) 1, 3, 4 ו-5

שאלה מס' 5 (10 נקודות)

מה תדפיס התכנית הבאה?

```
#include <stdio.h>
#include <stdlib.h>

int a = 6;
int b = 9;

void g(int *y)
{
  int b = 14;
  *Y += b;
}

void f(int *x)
{
  int b = 13;
  (*x)++;
  g(&a);
}

int main(void)
{
  int a = 11;
  f(&a);
  g(&b);
  printf(" a=%d  b=%d \n",a,b);
}
```

a) a=6 b=9
b) a=11 b=14
c) a=11 b=23
d) a=6 b=13
e) a=12 b=23

```
Node* build (char* string)
{
  int i, letters=0

  for (i=0; i<len; i++)
  {
    while ((strcmp(string[i], " ") != 0) && (i<len))
    {
      if (string[i+1] == " ")
      {
        node = create_node(string[i], letters);
      }
      else
      {
        node = create_node(string[i], 0);
      }
```

```
char * strstr(const char* str, const char * substr) {
    int i,j;
    char* p = NULL;
    int short = strlen(substr);
    int long = strlen(str);
    if (short > long)
        return p

    for (i=0; i<long; i++)
    {
        for (j=0; j<=short; j++)
            if (strcmp(str[i+j], str[j]) != 0) ?
                break;
        if (j == short)
            p = &str[i];
    }
    return (p)
}
```

- הוגן, strcmp לא נמצא (נורמלי)

X

$\frac{7}{12}$

```
{
    floor=floors(experiments-1, balls) +
          floors(experiments-1, balls-1) + 1;
    add(floor, balls, experiments);
    return(floor);
}
}
```

מחזירים המינימום - מינימום הקומה

add (0,0,0);

הפונקציה add - מוסיפה לרשימה חדשה לטבלה

הפונקציה search - מחפשת בתוך הטבלה.

בעיקרון 2 הפונקציות הללו משתמשות בטבלה שהוא יש הגבלה
של 2 ערכים שלמים, לכן ה floor הוא מקסימום
בגודל unsigned int ולכן שלא יגרום לבעיה
בגודל size שלו ולפרוץ מן הערכים

- חסר אתחול הנתון (-3

3)

26/30

```
#include<stdio.h>

int size = sizeof(unsigned int);
void* checks = Null;
int num = 0;
void add (unsigned int floor, unsigned int balls,
          unsigned int experiments);
{ int size;  num++;
  if (checks == Null)
  {
    checks = malloc (3*sizeof (unsigned int));
    *checks = experiments;
    *(checks+size) = balls;
    *(checks+2*size) = floor;
  }
  else
  { size = sizeof (checks);
    realloc (checks, sizeof(checks)+
                 3*sizeof (unsigned int));
    *checks+num*size        = experiments
    *(checks+(num+1)*size   = balls;
    *(checks+(num+2)*size   = floor;
  }
}

unsigned int search (unsigned int experiment, unsigned int
                     balls)
{ int i=0;
  while (i<=num)
  {
    if (*(checks+3*size*i) == experiment
        (*(checks+size*i+1) == balls
    {
      return (*checks+size*3*i+2)
    }
    i++;
  }
  return (-1)
}

unsigned int floors (unsigned int experiments,
                     unsigned int balls)
{
  unsigned int floor;
  if ((floor = search(experiments, balls)!=(-1))
    return floor;
```

מחברת מס' 2
מתוך 11 מחברות

TEL AVIV UNIVERSITY אוניברסיטת תל-אביב

19

לפני התחלת הבחינה מלא את כל הפרטים הבאים בכתב ברור וקרא בעיון את ההוראות:

1. על הנבחן להיבחן רק בחדר שבו הוא רשום.
2. עם הכניסה לחדר הבחינה יש להניח את החפצים בצד לרבות מכשירי קשר ואמצעי תקשורת אחרים כשהם כבויים.
3. אסור להחזיק בהישג יד, בחדר הבחינה או בסמוך לו, כל חומר הקשור לבחינה/לקורס פרט לחומר שהשימוש בו הותר בכתב על ידי המורה.
4. יש למלא את הפרטים על מחברת הבחינה במקום המיועד לכך בלבד. אין לכתוב את השם או כל פרט מזהה אחר בתוך המחברת.
5. יש להישמע להוראות המשגיח. נבחן לא יעזוב את מקומו ללא קבלת רשות המשגיח. הפונה בשאלה או בבקשה ירים את ידו.
6. נבחן שנכנס לחדר הבחינה וקיבל את השאלון (טופס הבחינה) לידו ייחשב כמי שנבחן במועד זה. היה והחליט לא לכתוב את הבחינה, לא יהא רשאי לעזוב את חדר הבחינה, אלא כעבור חצי שעה ממועד תחילתה ולאחר שהחזיר את המחברת והשאלון. ציונו בבחינה יהיה "0".
7. קריאת השאלון מותרת רק לאחר קבלת רשות המשגיח.
8. יש לכתוב את התשובות בעט, בכתב יד ברור ונקי. נבחן הבוחר לכתוב טיוטה יעשה זאת בעמודו הימני של דפי מחברת הבחינה ויציין בראש העמוד "טיוטה". אין לתלוש דפים מהמחברת.
9. מחברות הבחינה שקיבל הנבחן תהיינה בפיקוחו ובאחריותו במשך כל הבחינה. בעת יציאה מן החדר יופקדו המחברות והשאלון בידי המשגיח.
10. בתום הבחינה יחזיר הנבחן את המחברות והשאלון ויקבל מידי המשגיח את כרטיס הנבחן.
11. הנוהג בניגוד להוראות ול"נוהל סדרי בחינות ודיווח ציונים" צפוי להפסקת בחינתו ואף להעמדה לדין משמעתי.
12. אין לכתוב מעבר לקו האדום משני צידי הדף.

בהצלחה.

לשימוש המורה הבוחן:

הציון _____

המחברת נבדקה ביום _____

חתימת המורה _____

תאריך הבחינה 23/9

שם הקורס תורנה 1

שם המורה צפי פוגל + גיל נהון אור

החוג/המגמה מדעי המדינה

64033

(א)

// הפונקציה יוצרת node חדש שהו ה-char הוא c והפרמטר end נשלח
// אם מתבטל עליה זה אחרת.

```
Node * create_node (char c, int end)
{
    Node * node;

    node = (Node *) malloc (sizeof(Node));
    if (node == NULL) return Null;
    node->c = c;

    node->leftmost_child = Null;

    node->words = end;

    node->sibling = Null;

    return (node);
}
```

// הפונקציה מקבלת node ומחפשת בילדים שלו מילה שמתחילה באותו תו
// ברוכת שהרוב חוות מילו שהוא מחפש כלומר ה-sibling המשמאלי
// הנמוך כל עוד הוא קטן מהמילה שלו. אשר חוות לגוקיות אם יש לו
// הוא ומצביע לאותו המחרוזת. הפונקציה מחזירה מחרוזת יש הכלל השמאלי של הבן
// הבן מצביע לתחילת חוות המילה.

```
Node * search (Node * node, char c, int end)
{
    Node* new;
    if (node != NULL)
    {
        while (strcmp (node->c, c) <= 0)
        {
            node = node -> sibling
        }

        if (strcmp (node->c, c) != 0)
        {
            new = create_node (c, end);
            if (new == Null) return (Null);
            new->sibling = node->sibling;

            node->sibling = new;

            return (new -> leftmost_child)
        }
        else
        {
            node->word = word + end;
            return (node -> leftmost_child);
        }
    }
    else
    {
        node = create_node (c, end);
        if (node == Null) return Null;
        return (node -> leftmost_child);
    }

    return (Null);
}
```

הסיבוכיות עדיפה כאשר המילים מגיעות בסדר אלפבתי, היא תלויה ב המילה הארוכה ביותר בפרק, שזה כמובן המצב הטוב ביותר אבל לא ניתן לדעת מראש אם זה מה שיהיה.

```
Node* build (char* string)
{
    Node* root = Null;
    Node* node = Null;
    int i;
    int len = strlen(string);

    while (i<=len)
    {
        while (string[i] != " ")
        {
            if ((string[i-1] != " ") && (string[i+1] != "\0"))
            {
                if (root == Null)
                {
                    node = create_node(string[i],0);
                    if (node == Null) return (Null);
                }
                else
                {
                    node = search(node, string[i],0)
                }
                i++;
            }
            else
            {
                if (root == Null)
                {
                    node = create_node(string[i],1);
                    if (node == Null) return (Null);
                }
                else
                {
                    node = search(node, string[i], 1);
                }
                i++;
            }
        }
        i++;
        node = root;
    }
}
```

מכניסים בו ל root ומעדכנים. כי יכול להיות מצב שהרו הראשון לא יכנס כלל. (-1)

יש לחבר לשורש בתחילת כל מילה כי יכול להיות מצב שמעדכנים אותו ומפספסים את הראשון. (-2)

עדכון לפי מילה

מחזיר לשורש המילה הבאה

(א)

```
void frequent_words (Node * root)
{
    int numofwords = 0;        Node * node1 = root;   // מצביע לשורש
        numoftimes = 0;        Node * node2 = root;   // מצביע לשורש
    while (node1->sibling != NULL)
    {
        while (node2->leftmost_child != NULL)
        {
            if ((node2->words) > numoftimes))
            {
                numoftimes = node->words;   // מילה פעמים הכי הרבה
                numofwords = 1;             // מונה כמה מילים
            }
            else
            {
                if ((node2->words) == numoftimes))
                {
                    numofwords++;           // יש מילה פעמים כמה הכי
                }
            }
            node2 = node2->leftmost_child;
        }
        node1 = node->sibling;              // קדם יציב
        node2 = node1;                      // מתחיל מלמעלה שמילה לפני
    }
    printf("%d %d", numoftimes, numofword);
}
```

מתחיל מ- root אם אין לו siblings (-2)

הנשחק numofwords כמה מילים נפלו הכי הרבה פעמים
הנשחק numoftimes - שכיחות

18/20