מבוא למדעי המחשב – אוסף תרגילים ופתרונות

איסוף ועריכה: עילאי הנדין

תוכן עניינים

סמסטר ב' תשס"ה

סמסטר א' תשס"ה

סמסטר א' תשס"ג

מבוא למדעי המחשב

סמסטר ב' תשס"ה

תרגיל מס' 2

תאריך הגשה: 16/3/05

בדיקת תקינות תאריכים

כתבו תוכנית המקבלת כקלט נתונים המייצגים תאריך בלוח האזרחי, ובודקת האם התאריך הוא תאריך תקין. אם התאריך אינו תקין, על התכנית להדפיס הודעה המסבירה מדוע.

להלן רשימה של החודשים ומספר הימים שבכל חודש:

מס' חודש	שם חודש	מס' ימים
1	ינואר	31
2	פברואר	28 (29)
3	מרץ	31
4	אפריל	30
5	מאי	31
6	יוני	30
7	יולי	31
8	אוגוסט	31
9	ספטמבר	30
10	אוקטובר	31
11	נובמבר	30
12	דצמבר	31

מספר הימים בחודש פברואר בשנה מעוברת מסומן בסוגריים. שנה היא מעוברת אם היא מתחלקת ב-4 אך אינה מתחלקת ב-100, או שהיא מתחלקת ב-400 כפי שראינו בתרגול.

הקלט לתוכנית צריך להיות בפורמט הבא: *dd mm yyyy* כאשר: *dd , mm* ו- yyyy הם מספרים שלמים וחיוביים המייצגים יום, חודש ושנה בהתאמה. הטווחים המותרים הם:

ימים: $1 \le dd \le 31$

חודשים: $1 \le mm \le 12$

שנים: $0 \le mm \le 9999$

שימו לב כי אין זה מספיק שהמספרים נמצאים בטווחים המותרים על מנת שהתאריך יהיה תקין.

לדוגמא: ***31 4 2000*** - אינו תאריך תקין מכיוון שבאפריל יש 30 ימים. וכן הלאה...

כאמור, אם התאריך המתקבל כקלט הינו תאריך תקין, על התכנית להדפיס הודעה שהתאריך תקין. אם התאריך אינו תקין, על התכנית להדפיס הודעה שהתאריך אינו תקין ולציין את הסיבה.
ראו דוגמאות קלט/פלט באתר הקורס.

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

int main() {
        unsigned int dd;
        unsigned int mm;
        unsigned int yyyy;

        printf("Please input one date in the format:\ndd mm yyyy\n");

        if (3!=scanf("%u%u%u",&dd,&mm,&yyyy)) {
                printf("Problems reading input. Input may be invalid\n");
                exit(EXIT_FAILURE);
        }

        if (0>yyyy || 9999<yyyy) {
                printf("Invalid year %d. "
                        "You can only use years from 0000 to 9999\n",yyyy);
                exit(EXIT_FAILURE);
        }

        if (1>mm || 12<mm) {
                printf("Invalid month %d. "
                        "You can only use months from 01 to 12\n",mm);
                exit(EXIT_FAILURE);
        }

        if (1>dd || 31<dd) {
                printf("Invalid day %d. "
                        "You can only use days from 01 to 31\n",dd);
                exit(EXIT_FAILURE);
        }


        if (31==dd && (11==mm || 9==mm || 6==mm || 4==mm)) {
                printf("The %dth month cannot have %d days\n",mm,dd);
        } else if (29<dd && 2==mm) {
                printf("The %dnd month cannot have %d days\n",mm,dd);
        } else if (28<dd && 2==mm && (!(yyyy%4==0&&(!((yyyy%100==0)&&(yyyy%400!=0)))))) {
                printf("The %dnd month has %d days only on leap years, and %d is "
                        "not a leap year.\n",mm,dd,yyyy);
        } else {
                printf("%d/%d/%d is a valid date\n",dd,mm,yyyy);
        }

        return EXIT_SUCCESS;
}
```

מבוא למדעי המחשב

סמסטר ב' תשס"ה

תרגיל מס' 3

תאריך הגשה: 24/3/05

.1

כתבו תכנית המבצעת פעולות שונות על מספרים שלמים וחיוביים כדלקמן:
התוכנית תקלוט מספר, ותציג את תפריט הפעולות הבא למשתמש:

(1) - חשב את מספר הספרות של המספר. (עבור 23567 , התוצאה תהיה 5)
(2) - בדוק אם המספר הוא מספר ראשוני והדפס הודעה למשתמש אם המספר הוא ראשוני או לא. מספר הוא ראשוני אם הוא מתחלק בעצמו ובאחד בלבד (חוץ מהמספר 1).
(3) - יציאה

בשלב זה, יש לבקש מהמשתמש לבחור באחת מהאפשרויות, לבצע את החישוב הנדרש, ולהדפיס את תוצאת החישוב.
התוכנית תמשיך לקבל קלט מהמשתמש ותציג את התפריט, עד שהמשתמש יבחר באפשרות של יציאה.
במקרה של קלט שאינו בטווח הרצוי, יש להמשיך לבקש קלט מהמשתמש עד לקבלת קלט רצוי. ראו דוגמאות קלט/פלט באתר הקורס.

.2

כתבו תוכנית הקולטת מספר חיובי שלם שאינו מכיל את הספרה 0 (יש לוודא זאת בתוכנית), בודקת אם המספר הוא פלינדרום, ומדפיסה למסך את התשובה.
מספר הוא פלינדרום אם ניתן לקרוא אותו גם משמאל לימין וגם מימין לשמאל.
לדוגמא: 2332, 579975, 121 , 9, 8888 ....
אין להשתמש במחרוזות או במערכים לצורך הפיתרון. יש לקלוט את המספר כ- integer .
ראו דוגמאות קלט/פלט באתר הקורס.

```
#include <stdio.h>

int main()
{

        int num, i, option=-1, counter;


        while(option != 3) {      /* keep presenting menu untill exit */

                do{

                        printf("Enter a positive number: ");
                        scanf("%d",&num);

                }while(num<=0);

                printf("Select one option from the following:\n\n");
                printf("(1) calculate the number of digits of %d.\n",num);
                printf("(2) check if the number %d is a prime number.\n",num);
                printf("(3) Exit.\n");

                do{
                        printf("Enter option:");
                        scanf("%d",&option);
                }while(option<1 || option>3);

                switch(option) {

                        case 1:
                                counter=0;
                                while(num>0) {
                                        num/=10;
                                        counter++;
                                }
                                printf("\nThe number of digits is %d.\n",counter);
                                break;

                        case 2:
                                if(num==1){
                                        printf("\nThe number is not a prime
number.\n");
                                        break;
                                }
                                for(i=2 ; i<num ; i++){
                                        if(num%i==0){
                                                printf("\nThe number %d is not a prime
number.\n",num);
                                                break;
                                        }
                                }
                                if(i==num)
                                        printf("\nThe number %d is a prime
number.\n",num);
                                break;

                        case 3:
                                printf("Exit.\n");
                                return 0;
                }
```

```
                printf("\n\n");
        }
}
```

```
#include <stdio.h>

int main()
{
        int num, copy_num;
        int left, right;
        int i,counter, contain_0;

        do{
                printf("Enter number: ");
                scanf("%d",&num);
                copy_num=num;
                contain_0=0;
                while(copy_num>0){
                        if((copy_num%10)==0){
                                contain_0=1;
                                break;
                        }
                        copy_num/=10;
                }

        }while(num<=0 || contain_0);

        copy_num=num;

        while(copy_num>10){ /* keep checking if we steel have 2 digits or more */

                left=copy_num;
                right=copy_num;

                counter=0;
                while(left>10) {     /* get the left digit */
                        left/=10;
                        counter++;
                }

                right=copy_num%10;   /* get the right digit */

                if(right != left) {
                        printf("The number %d is not a palindrome\n",num);
                        return 0;
                }

                for(i=0; i<counter ; i++)  /* calculate what is number to be reduced
from the left side */
                        left*=10;

                copy_num= (copy_num -(left + right))/10; /* calculate what is the new
number to check */

        }

        printf("The number %d is a palindrome\n",num);
        return 0;
}
```

מבוא למדעי המחשב

סמסטר ב' תשס"ה

תרגיל מס' 4

תאריך הגשה: 31/3/05

.1

כתבו תוכנית הבודקת את מספר הפעמים בו מופיע רצף התווים abc במחרוזת קלט כלשהיא. את הקלט יש לקבל מקובץ באמצעות redirection . לדוגמא, עבור הקלט : ab?yabcabb*caabcc

התוכנית תדפיס : The string abc occurs 2 times.

.2

בהינתן סדרת מספרים, **חץ** הוא תת-סדרה רציפה עולה.

למשל, בסדרה : 1,6,12,-2,3,5,-8,0,3,10 תת-הסדרה 1,6,12 היא חץ, ותת הסדרה 8,0,3,10-, היא החץ הארוך ביותר.

תת-הסדרה 5,-8,0 אינה חץ (כי אינה עולה) והמספרים 1,3,5 אינם חץ כי אינם תת-סדרה רציפה.

כתבו תוכנית הקוראת מספרים שלמים מהקלט, עד לסוף הקלט, ומדפיסה את אורך החץ הארוך ביותר בסדרה ואת מיקום ההתחלה שלו (כלומר, את אינדקס האבר הראשון בחץ). ניתן להניח שבקלט לפחות אבר אחד, ושהקלט תקין (כלומר כולל מספרים שלמים בלבד).

למשל, אם הקלט הוא : 10 3 0 8- 5 3 2- 12 6 1 תדפיס התוכנית :

The longest arrow is of length 4, starting at position 7.

את הקלט יש לקבל מקובץ באמצעות redirection .

<u>הערה חשובה</u> : אין להשתמש במערכים ומחרוזות בתרגילים הללו.

```c
/* count-abc.c */
/* This program counts the number of occurrences of the string "abc" in its input */

#include <stdio.h>

int main ()
{
  int c;
  int count=0;
  char state=0;

  while ((c=getchar()) != EOF) {
    if (c == 'a') {
      state ='a';
    } else if (state == 'a' && c == 'b') {
      state = 'b';
    } else if (state == 'b' && c == 'c') {
      count++; state = 0;
    } else {
      state = 0;
    }
  }
  printf ("The string %s occurs %d times.\n", "abc", count);
  return 0;
}

/* arrow.c */
/* Read numbers until EOF, print the length of the longest
   increasing (contiguous) sub-sequence */

#include <stdio.h>

int main ()
{
  int input1,input2,length=1,max_length=1,start=1,max_start=1,i=1;

  scanf ("%d", &input1); /* the input is valid and has at least one
                            number */
  while (scanf("%d",&input2)==1) { /* while not end of file */
    i++; /* the index of the current number */
    if (input2>input1) { /* still monotone */
      length++;
    } else { /* breaking a monotone sequence */
      length=1; /* a new sequence starts here, its length is 1 */
      start=i;
    }
    if (length>max_length) { /* if we're currently breaking a record */
      max_length=length; /* update records */
      max_start=start;
    }
    input1=input2; /* prepare to read new input */
  }
  printf ("The length of the longest arrow is %d, "
          "starting from the %d-th element\n",max_length,max_start);
  return 0;
}
```

מבוא למדעי המחשב

סמסטר ב' תשס"ה

תרגיל מס' 5

תאריך הגשה: 7/4/05

1. הגפרור האחרון

עליכם לממש משחק מחשב בשם "הגפרור האחרון". המשחק מתנהל בין שני שחקנים, כאשר נתונים : מספר גפרורים הנמצאים על שולחן המשחק , ומספר מקסימלי של גפרורים שניתן להוריד.
על השחקנים לשלוף, כל אחד בתורו, לפחות גפרור אחד מהשולחן ולא יותר מהמספר המקסימלי הנתון. המנצח הוא מי ששולף את הגפרור האחרון (או הגפרורים האחרונים) שנותר על השולחן .

עליכם לממש תוכנית הקולטת מהמשתמש את מס' המשחקים הרצוי, לנהל את המשחקים, ולבסוף להדפיס כמה משחקים ניצח כל שחקן.
בכל משחק יש לבקש כקלט את מספר הגפרורים ההתחלתי, את המספר המקסימלי של גפרורים שניתן להוריד בכל תור, מי מהשחקנים מעוניין להתחיל לשחק, ולאחר מכן יש להציג את שולחן המשחק. במהלך המשחק יש להדפיס תורו של מי כעת, ולקלוט מאותו שחקן מספר חוקי של גפרורים להורדה מהשולחן, כל עוד נשארו גפרורים. יש להדפיס בכל שלב את שולחן המשחק עם מספר הגפרורים הנוכחי. בסוף כל משחק יש להדפיס מי המנצח. כאמור , מי ששולף את הגפרור האחרון הוא המנצח .

יש לממש את הפונקציות הבאות :

RemoveMatches – הפונקציה מקבלת כפרמטרים את מספר הגפרורים הנוכחי ואת המספר המקסימלי של גפרורים, ותקלוט מהמשתמש מס' גפרורים להורדה מהשולחן. אם הקלט חוקי (חישבו מתי!) הפונקציה תחזיר את מס' הגפרורים שלאחר ההורדה, אחרת תחזיר 1- .

ShowTable - הפונקציה תדפיס את מס' הגפרורים הנוכחי, (יש להדפיס גם ציור מלבד המספר עצמו, לדוגמא : עבור 5 גפרורים יש לצייר |||||) את המספר המקסימלי להורדה, ותורו של מי כעת.

GetNextPlayer – פונקציה שבכל קריאה אליה תעביר את התור לשחקן השני. הפונקציה תחזיר מי משהחקנים תורו כעת.

ManageGame – פונקציה שמנהלת משחק בודד. הפונקציה תקלוט מהמשתמש את הנתונים למשחק כפי שפורט לעיל, עד לקבלת קלט תקין בטווח הרצוי, תנהל את המשחק, ותדפיס בכל שלב את מצב השולחן. בסוף המשחק יש להדפיס מי הוא המנצח. הפונקציה תחזיר ערך שמשמעותו מי מהשחקנים ניצח.

כאמור, יש לנהל את מספר משחקים הרצוי ולהדפיס לבסוף את התוצאות הסופיות.
יש לכתוב לפני כל פונקציה הערה שמתארת מה מבצעת הפונקציה, מה הארגומנטים שהיא מקבלת ואיזה ערך היא מחזירה.

יש לחלק את התוכנית לקובץ כותרים המכיל את ההצהרות (קובץ h.) , קובץ המכיל את הגדרות הפונקציות (קובץ c.) וקובץ המכיל את פונקצית ה- main המפעילה את התכנית. יש לבצע קומפילציה בעזרת make כפי שנלמד בתרגול. (קובץ ה- Makefile נמצא באתר הקורס).

ראו דוגמאות קלט/פלט באתר הקורס.

```
/* matches.h */

#ifndef _MATCHES_H
#define _MATCHES_H

int RemoveMatches(int, int);
void ShowGame(int,int,int);
int GetNextPlayer(int);
int ManageGame();


#endif

/* matches.c */

#include <stdio.h>
#include "matches.h"

int RemoveMatches(int matches, int max_matches)
{
        int remove;

        printf("Put number of matches to remove:\n");
        scanf("%d",&remove);

        if(remove > max_matches || remove <= 0 || remove > matches)
        {
                printf("Wrong input. Try again\n");
            return -1;
        }

        return matches - remove;
}


void ShowGame(int matches, int max_matches, int turn)
{
        int i;
        printf("The number of matches now in the table is: %d\n",matches);
        printf("The maximal number of matches that can be removed is:
%d\n",max_matches);

        for (i=0 ; i<matches*2 ; i++)
                printf("-");
        printf("\n");

        for (i=0 ; i<matches ; i++)
                printf("| ");
        printf("\n");

        for (i=0 ; i<matches*2 ; i++)
                printf("-");
        printf("\n");

        if(turn == 1)
                printf("** Player 1 it is your Turn **\n");
        else
                printf("** Player 2 it is your Turn **\n");
}

int GetNextPlayer(int turn)
```

```
{
        if(turn == 1)
                return 2;
        else
                return 1;
}

int ManageGame()
{
        int matches, current_matches;
        int max_matches;
        int turn;

        do{
                printf("Enter number of matches:\n");
                scanf("%d",&matches);
        }while (matches <=0);

        do{
            printf("Enter the maximum number of matches: (1 - %d)\n",matches);
                scanf("%d",&max_matches);
        }while ((max_matches <= 0) || (max_matches > matches));

        do{
                printf("Which player wants to start? press 1 or 2.\n");
            scanf("%d",&turn);
        }while ((turn != 1) && (turn != 2));


        while(matches > 0)
        {
                ShowGame(matches, max_matches, turn);

                do{
                        current_matches = RemoveMatches(matches,max_matches);
                }while(current_matches == -1);

                if( (matches = current_matches) != 0)
                    turn = GetNextPlayer(turn);
        }

        if(turn == 1)
        {
                printf("player 1 won the game!\n");
                return 0;
        }
        else
        {
                printf("player 2 won the game!\n");
                return 1;
        }
}
```

```c
/* Main */

#include <stdio.h>
#include "matches.h"

int main()
{
        int games;
        int i;
        int win1 = 0;
        int win2 =0;

        printf("How many games do you want to play?\n");
        scanf("%d",&games);

        for (i=1 ; i <= games ; i++)
        {
                printf("**** GAME NUMBER %d ****\n",i);
                if(ManageGame())
                        win2++;
                else
                        win1++;
        }
        printf("!!!!!! The final score !!!!!!:\n");
        printf("Player 1 won %d games.\n",win1);
        printf("Player 2 won %d games.\n",win2);

        return 0;
}
```

מבוא למדעי המחשב

סמסטר ב' תשס"ה

תרגיל מס' 6

תאריך הגשה: 20/4/05

חיפוש בטקסט

כתבו תוכנית המקבלת כקלט מחרוזת וטקסט ומבצעת את אחת מהאפשרויות הבאות:

א. הדפסת כל השורות בטקסט בהם מופיעה המחרוזת.

ב. הדפסת כל המילים בטקסט הדומות למחרוזות עד כדי השמטת אות אחת מהמילה.

לדוגמא אם אנו מחפשים את המילה 'cat' ונתון הטקסט:

```
this is a text file
looking for the word cat
the program should print also cats
after cats crat and lcat are printed
the program shouldn't print
the word caats
```

התכנית תדפיס עבור אפשרות ב' את המילים:

```
cat
cats
cats
crat
lcat
```

ועבור אפשרות א' תדפיס:

```
looking for the word cat
the program should print also cats
after cats crat and lcat are printed
```

מכיוון שהמחרוזת 'cat' מופיעה בשורות הללו.

<u>הערה</u>: המילים בטקסט מופרדות באחד מהתווים הבאים: ' ' (space) '\t' (tab) '\n' (newline) וכל שורה מסתיימת בתו '\n' (newline).

חלוקה לפונקציות

עליכם לממש את הפונקציות הבאות:

int getline(char s[], int lim)

הפונקציה מקבלת מחרוזת s וערך lim (שהוא הגודל המקסימלי של תווים שהמחרוזת יכולה להכיל עפ"י הגדרתה), קולטת שורה (תו אחר תו) מה- standard input, ומאחסנת אותה במחרוזת s. יש לוודא שלא מתבצעת חריגה מהערך lim. לבסוף הפונקציה תחזיר את מספר התווים שקלטה.

יש להגדיר את גודל השורה כקבוע בתוכנית: #define LINE 256

int getword(char w[], int lim)

הפונקציה מקבלת מחרוזת w וערך lim (שהוא הגודל המקסימלי של תווים שהמחרוזת יכולה להכיל עפ"י הגדרתה), קולטת מילה (תו אחר תו) מה- standard input, ומאחסנת אותה במחרוזת s. יש לוודא שלא מתבצעת חריגה מהערך lim. לבסוף הפונקציה תחזיר את מספר התווים שקלטה. כזכור המילים מופרדות ע"י התווים הבאים: ' ' (space), '\t' (tab), '\n' (newline).

יש להגדיר את גודל המחרוזת כקבוע בתכנית: #define STR 30

int substring(char * str1, char * str2)

הפונקציה מקבלת שתי מחרוזות str1 ו- str2 ובודקת האם str2 מוכלת ב- str1 (תת מחרוזת).
לדוגמא: aba היא תת מחרוזת של aagabahj. הפונקציה תחזיר 1 אם כן ו-0 אם לא.

int similar (char *s, char *t, int n)

הפונקציה מקבלת שתי מחרוזות s ו- t ומספר n ובודקת את הדמיון בין המחרוזות באופן הבא:
הפונקציה בודקת האם ניתן להגיע מהמחרוזת t למחרוזת s ע"י השמטה של n תווים מ-t.
אם כן הפונקציה תחזיר 1, אחרת תחזיר 0.
לדוגמא אם נתון ש-t היא המחרוזת "csintro" ו- s היא המחרוזת "cintr" ו- n הוא 2
הפונקציה תבדוק האם ניתן להוריד שני תווים מ-t לקבל את s. עפ"י הדוגמא לעיל הפונקציה תחזיר 1 מכיוון שאפשר להשמיט את התווים s ו- o מ-t לקבל את s.
עבור המחרוזות "csintro" ו- "csintro" ומספר 0, הפונקציה תחזיר 1 שכן בהשמטת 0 תווים ניתן להגיע מ- s ל- t.

void print_lines(char * str)

הפונקציה מקבלת את המחרוזת הרצויה לחיפוש, קולטת את שורות הטקסט, ומדפיסה את השורות בהם מופיעה המחרוזת, תוך שימוש בפונקציות הרלוונטיות שהוגדרו לעיל.

void print_similar_words(char * str)

הפונקציה מקבלת את המחרוזת הרצויה לחיפוש, קולטת את מילות הטקסט ומדפיסה את המילים הדומות למחרוזת החיפוש עד כדי השמטה של אות אחת מהמילים המופיעות בטקסט (כולל מילים הזהות למחרוזת החיפוש).
הפונקציה תשתמש בפונקציות הרלוונטיות שהוגדרו לעיל.

קלט

את הקלט יש לקבל מקובץ באמצעות redirection (./find < input.txt).
הפורמט של קובץ הקלט ייראה באופן הבא:

```
cat a

this is a text file
looking for the word cat
the program should print also cats
and crat and lcat but it shouldn't print
```

בשורה הראשונה תופיע מחרוזת החיפוש (cat) ולאחריה אפשרות החיפוש (a או b)
כאשר a מסמנת הדפסת שורות בהם מופיעה המחרוזת cat ו- b הדפסת מילים הדומות ל-cat עד כדי השמטה של אות אחת. לאחר מכן, החל מהשורה ה-3 יופיע הטקסט לחיפוש.
ראו דוגמאות קלט/פלט באתר הקורס.
יש להקפיד שהקלט/פלט יהיה זהה לחלוטין לדוגמאות הניתנות באתר הקורס.

```
/* find.h */

#ifndef _FIND_H_
#define _FIND_H_

#define LINE   256
#define WORD   30

int getline(char s[], int lim);
int getword(char s[], int lim);
int substring( char * str1, char * str2);
int similar(char *s, char *t, int n);
void print_lines(char * str);
void print_similar_words(char * str, int n);

#endif
```

/* find.c */

```
#include <stdio.h>
#include <string.h>
#include "find.h"

int getline(char s[], int lim)
{
  int c, i, j;

  for(i = 0, j = 0; (c = getchar())!=EOF && c != '\n'; ++i)
  {
    if(i < lim - 1)
    {
      s[j++] = c;
    }
  }
  if(c == '\n')
  {
    if(i <= lim - 1)
    {
      s[j++] = c;
    }
    ++i;
  }
  s[j] = '\0';
  return i;
}

int getword(char s[], int lim)
{
  int c, i, j;

  for(i = 0, j = 0; (c = getchar())!=EOF && c != '\n'&& c != '\t'&& c != ' '; ++i)
  {
    if(i < lim - 1)
    {
      s[j++] = c;
    }
  }

  s[j] = '\0';
```

```
  return i;
}

int substring( char * str1, char * str2)
{
  char *cp = str1;
  char *s1, *s2;
  if ( !*str2 )
    return 1;
  while (*cp) {
    s1 = cp;
    s2 = str2;
    while ( *s1 && *s2 && !(*s1-*s2) )
      s1++, s2++;
    if (!*s2)
      return 1;
    cp++;
  }
  return 0;
}


int similar(char *s, char *t, int n) {

        int i;

        while(*s) {

                i=0;
                while(*(t+i) != *s) {
                        if(!(*(t+i)))
                                return 0;
                        i++;
                }
                n-=i;
                s=s+1;
                t=t+i+1;
        }
        return (n==strlen(t));
}

void print_lines(char * str) {

        char line[LINE];

        while(getline(line,LINE)) {

                if(substring(line,str))
                        puts(line);
        }
}
```

```
void print_similar_words(char * str, int n) {

        char word[WORD];
        int i;

        while(getword(word,LINE)) {
                i=n;

                while(i>=0) {
                        if(similar(str,word,i))
                                printf("%s\n",word);
                        i--;

                }
        }
}
```

/* main */

```
#include <stdio.h>
#include <string.h>
#include "find.h"

int main()
{
        char word[WORD];
        char option;
        getword(word,WORD);
        option=getchar();

        getchar();
        getchar();

        if(option!='a' && option !='b'){
                printf("Bad selection.\n");
                exit(1);
        }

        if(option=='a') {
                printf("Printing lines that contain the word '%s':\n\n",word);
                print_lines(word);
        }

        if(option=='b') {
                printf("Printing the similar words for '%s' up to 1 degree of
similarity:\n",word);
                print_similar_words(word,1);
        }
}
```

מבוא למדעי המחשב
סמסטר ב' תשס"ה

תרגיל מס' 7
תאריך הגשה: 1/5/05

תפזורת מילים

בתרגיל זה תממשו משחק תפזורת מילים, כאשר נתונה טבלה דו-מימדית של אותיות ובתוכה מסתתרות מילים שונות. המילים יכולות להסתתר בטבלה ב- 4 אופנים:
1) משמאל לימין. (--)
2) מלמעלה למטה. (|)
3) מלמעלה למטה באלכסון ימני. (\ )
4) מלמעלה למטה באלכסון שמאלי. (/)

בשלב הראשון עליכם לקלוט את המילים, לשזור אותם בטבלה, למלא את המקומות הריקים בטבלה באותיות אקראיות, ולהציג את טבלת התפזורת ורשימת המילים שעל המשתמש למצוא.
הנחת עבודה: המילים הנקלטות מכילות אותיות קטנות בלבד (lower-case). על-כן יש למלא את הטבלה באותיות אקראיות קטנות. (נשתמש באותיות גדולות לצורך הדגשה כפי שיפורט בהמשך).
בשלב השני, יש לחפש את המילים בתפזורת. לגבי כל מילה יש לקלוט את הקואורדינטות של האות הראשונה שלה בתפזורת. אם מיקום המילה (באחד מ-4 אופני ההסתרה השונים) נמצא עבור קואורדינטות מסויימות, תודפס הודעת הצלחה עבור אותה מילה. אחרת, תודפס הודעת כישלון.
בסוף המשחק, לאחר מציאת כל המילים בתפזורת, תודפס תוצאת מספר ההצלחות במציאת המילים לעומת סך כל הנסיונות.
בכל שלב יודפס המצב הנוכחי בטבלה כאשר המילה שנמצאה בתפזורת תודגש בתוך הטבלה, ורשימת המילים החדשה תכיל רק את המילים שטרם נמצאו.

יש לממש את הפונקציות הבאות:

1) `void init_table(char Table[SIZE][SIZE])`

הפונקציה תקבל את טבלת המשחק ותאתחל את כל איבריה ב- '0', כאשר גודל הטבלה מוגדר בראש התוכנית ב- #define לדוגמא: `#define SIZE 15`

2) `int get_words(char words[WORDS_NUM][LENGTH])`

הפונקציה תקלוט מהמשתמש את המילים הרצויות לשיבוץ בתפזורת ותאחסן אותם ב- `words[WORDS_NUM][LENGTH]` כאשר WORDS_NUM ו- LENGTH הם קבועים המסמנים את מספר המילים המקסימלי ואורך מקסימלי של כל מילה בהתאמה. הקבועים יוגדרו בראש התוכנית לדוגמא: `#define WORDS_NUM 15` ו- `#define LENGTH 10`.
על הפונקציה לוודא שמתקבל קלט חוקי כגון: מספר מילים מתאים ואורך מילה בטווח של גבולות הטבלה. הפונקציה תחזיר את מספר המילים שנקלטו.

3)
```
void show_table(char Table[SIZE][SIZE],char words[WORDS_NUM][LENGTH],
                int flags[WORDS_NUM],int num_words)
```

הפונקציה תדפיס את הטבלה הנוכחית, ואת רשימת המילים שטרם נמצאו. המערך flags מכיל ערכים בינארים עבור רשימת המילים, כאינדיקציה לכך אם המשתמש מצא את מיקומם או לא. המיקום ה- i במערך מתאים למילה ה- i ברשימת המילים.

4)
```
int put_word(char Table[SIZE][SIZE],char word[])
```

על הפונקציה לשבץ את המילה word בתוך הטבלה החל ממקום כלשהו, באחד מארבע צורות השיבוץ שהוזכרו לעיל, בהנחה שהדבר מתאפשר. אם הפונקציה לא הצליחה לשבץ את המילה, יש לבחור נקודת התחלה אחרת. השיטה לבצע זאת היא כדלהלן:

1. שמרו בתוך משתנה tries את מספר הניסויים לשיבוץ שבצעתם.
2. הגרילו ערכי x ו-y למיקום התחלתי של נסיון השיבוץ, בעזרת הפונקציה rand().
3. הגרילו כיוון התקדמות (-- | \ /) . (לדוגמא מספרים מ-0 ועד 3) .
4. בדקו האם ניתן לשבץ את המילה עפ"י הערכים שהוגרלו. (אותיות מתנגשות, מיקום...). הבדיקה תתבצע באמצעות פונקציה 5.
   a . אם הצלחתם שבצו את המילה והחזירו את ערך כיוון השיבוץ.
   b . אם לא הצלחתם:
      I. קדמו את tries .
      II. אם tries < SIZE*SIZE . חזרו לשלב 2.
      III. אחרת, החזירו 1- (הפונקציה לא הצליחה לשבץ את המספר).

שימו לב כי קיימת אפשרות שהפונקציה תחזיר 1- על אף שניתן למצוא למילה שיבוץ בתוך התפזורת.
ניתן להשתמש בפונקציה srand() (מהספרייה stdlib) על מנת לקבל סדרה שונה של מספרים אקראיים בכל הרצה. ראשית יש לכלול את הספרייה time.h ולבצע את הפקודה srand(time(0)); בקובץ ה- main, לפני הקריאה לפונקציות. (אין חובה לעשות זאת).

5)
```
int check_location(char Table[SIZE][SIZE],char word[],int x,int y,int
                   direction)
```

הפונקציה תבדוק עבור מילה מסוימת, נקודת התחלה וכיוון, האם ניתן לשבץ את המילה. הפונקציה תחזיר 1 אם השיבוץ ניתן, אחרת היא תחזיר 0.

6)
```
int assign_words(char words[WORDS_NUM][LENGTH] ,int num_words)
                 Table[SIZE][SIZE], char
```

הפונקציה תשבץ את רשימת המילים בתפזורת ע"י קריאה לפונקציה put_word . לאחר שיבוץ כל המילים, הפונקציה תמלא את המקומות הריקים בתפזורת באותיות אקראיות. אם הפונקציה לא הצליחה לשבץ את כל המילים, היא תחזיר 1. אחרת, היא תחזיר 0.

7)
```
int find_word(char Table[SIZE][SIZE], char word[],int x,int y)
```

הפונקציה תבדוק עבור מילה מסויימת וקואורדינטות, האם המילה נמצאת בתפזורת החל מאותו מיקום באחד מכיווני השיבוץ השונים (-- | \ /). אם המילה נמצאה, הפונקציה תחזיר את הכיוון בו משובצת המילה, אחרת, תחזיר 1- .

8)

```
id highlight_solution(char Table[SIZE][SIZE], char word[],int x, int y,
                                int direction)
```

הפונקציה תדגיש מילה פתורה בטבלת התפזורת. הפונקציה תקבל מילה שנמצאה, מיקום האות הראשונה וכיוון השיבוץ, ותדגיש את המילה ע"י הפיכת ה- lower case ל- upper case . לדוגמא: אם המילה class נמצאת ברשימת המילים והמשתמש ניחש את מיקומה אזי היא תופיע מעתה בטבלת התפזורת כ- CLASS .

הערה: יש לקחת בחשבון את המקרה בו האותיות המודגשות הם חלק ממילה אחרת ברשימה.

9)

```
int get_all_locations(char Table[SIZE][SIZE],char
                                words[WORDS_NUM][LENGTH],
int num_words)
```

הפונקציה תציג את הטבלה הנוכחית ואת רשימת המילים העדכנית בכל שלב, ותקלוט מהמשתמש את מספר המילה אותה הוא מעוניין למצוא. הפונקציה תקלוט מהמשתמש את הקואורדינטות עבור המילה, תבדוק האם המילה נמצאת באותו מיקום, ותדפיס הודעה מתאימה. בשלב הבא, תוצג התפזורת ובה המילים הפתורות מודגשות כפי שהוסבר לעיל, ורשימת מילים עדכנית לפיתרון. וכך המשחק יתנהל עד שכל המילים תימצאנה, או עד שהמשתמש יקיש קואורדינטות שליליות.
לאחר מציאת כל המילים הפונקציה תדפיס את מספר ההצלחות במציאת המילים לעומת מספר הנסיונות הכולל .

- יש לחלק את התוכנית לשלושה קבצים: קובץ הצהרות (tifzoret.h), קובץ הגדרות, (tifzoret.c) וקובץ הפעלה שיכיל את פונקצית ה- main (tifzoret_main.c) כפי שראיתם בכיתה. את הקומפילציה יש לבצע באמצעות הפקודה make ב- Unix , אך לפני כן יש להוריד את ה- Makefile מאתר הקורס.
- ניתן להגדיר פונקציות נוספות עפ"י הצורך.
- ראו דוגמאות קלט/פלט באתר הקורס .

```
/* tifzoret.h */

#ifndef _TIFZORET_H_
#define _TIFZORET_H_


#define SIZE 15
#define WORDS_NUM 15
#define LENGTH 10

void init_table(char Table[SIZE][SIZE]);
int get_words(char words[WORDS_NUM][LENGTH]);
void show_table(char Table[SIZE][SIZE], char words[WORDS_NUM][LENGTH],int
flags[WORDS_NUM], int num_words);
int put_word(char Table[SIZE][SIZE],char word[]);
int check_location(char Table[SIZE][SIZE],char word[],int x,int y,int direct);
int assign_words(char Table[SIZE][SIZE], char words[WORDS_NUM][LENGTH],int
num_words);
int find_word(char Table[SIZE][SIZE], char word[],int x,int y);
void highlight_solution(char Table[SIZE][SIZE], char *word,int x, int y, int
direction);
int get_all_locations(char Table[SIZE][SIZE],char words[WORDS_NUM][LENGTH],int
num_words);

#endif
```

/* tifzoret.c */

```
#include<stdio.h>
#include<string.h>   /* for strlen() */
#include<stdlib.h>   /* for rand() */
#include"tifzoret.h"

int get_words(char words[WORDS_NUM][LENGTH])
{
        int num_words,i;
        do{
                printf("How many words do you want to enter? (1 - %d)\n",WORDS_NUM);
                scanf("%d",&num_words);
        }while(num_words>WORDS_NUM|| num_words<=0);

        for(i=0;i<num_words;i++)
        {
           do{
                   printf("Enter word (1-%d charactes): ",LENGTH -1);
                   scanf("%s",words[i]);

                   if(strlen(words[i]) > SIZE)
                        printf("Wrong input\n");
           }while(strlen(words[i]) > SIZE);
        }

        return num_words;
}
```

```
void init_table(char Table[SIZE][SIZE])
{
        int i,j;

        for(i=0;i<SIZE;i++)
      for(j=0;j<SIZE;j++)
                Table[i][j]='\0';
}

void show_table(char Table[SIZE][SIZE],char words[WORDS_NUM][LENGTH],int
flags[WORDS_NUM],int num_words)
{
          int i,j;
          printf("\n");

          for(i=0 ; i<SIZE ; i++)
      {
               for(j=0 ; j<SIZE ; j++)
              printf("[%c]",Table[i][j]);
            printf("\n");
      }

          printf("\nFind the following words:\n");

          for(i=0;i<num_words;i++)
            if(flags[i]==0)
                        printf("(%d) %s\n",i+1,words[i]);

          printf("\n");
}
```

```
int put_word(char Table[SIZE][SIZE],char word[])
{
        int direction, tries=0;
        int x,y,i;

        while(tries < SIZE*SIZE)
        {
                        x=rand()%SIZE;
                        y=rand()%SIZE;
                direction=rand()%4;

                        switch(direction)
                        {
                        case 0:           /* for the right */
                                if(check_location(Table,word,x,y,0)==1)
                                {
                                        for(i=0;i<strlen(word);y++,i++)
                                                Table[x][y]=word[i];
                                        return 0;    /* returns '0' in case of success
*/
                                }
                                break;
                        case 1:                 /* for down */
                                if(check_location(Table,word,x,y,1)==1)
                                {
                                        for(i=0;i<strlen(word);x++,i++)
                                                Table[x][y]=word[i];
                                        return 1;
                                }
                                break;
                        case 2:                 /* for right diagonal */
                                if(check_location(Table,word,x,y,2)==1)
                                {
                                        for(i=0;i<strlen(word);y++,x++,i++)
                                                Table[x][y]=word[i];
                                        return 2;
                                }
                                break;
                        case 3:                /*for left diagonal */
                                if(check_location(Table,word,x,y,3)==1)
                                {
                                        for(i=0;i<strlen(word);y--,x++,i++)
                                                Table[x][y]=word[i];
                                        return 3;
                                }
                        }
                        tries++;
        }

        return -1;         /* in case of failure */
}


int check_location(char Table[SIZE][SIZE],char word[],int x,int y,int direct)
{
        int i;
        switch(direct)
        {
                case 0:
                                if(y+strlen(word)>=SIZE)
                                        return 0;
```

```
                        for(i=0;i<strlen(word);y++,i++)
                        {
                                if(Table[x][y]!='\0' && Table[x][y]!=word[i])
                                        return 0;
                        }

                        break;

            case 1:
                        if(x+strlen(word)>SIZE)
                                return 0;
                        for(i=0;i<strlen(word);x++,i++)
                        {
                                if(Table[x][y]!='\0' && Table[x][y]!=word[i])
                                        return 0;
                        }
                        break;

            case 2:
                        if(y+strlen(word)>SIZE || x+strlen(word)>SIZE)
                                return 0;
                        for(i=0;i<strlen(word);y++,x++,i++)
                        {
                                if(Table[x][y]!='\0' && Table[x][y]!=word[i])
                                        return 0;

                        }
                        break;

            case 3:
                        if(strlen(word)>y || x+strlen(word)>SIZE)
                                return 0;
                        for(i=0;i<strlen(word);y--,x++,i++)
                        {
                                if(Table[x][y]!='\0' && Table[x][y]!=word[i])
                                        return 0;
                        }
        }
        return 1;  /*for success */
}
```

```
int assign_words(char Table[SIZE][SIZE], char words[WORDS_NUM][LENGTH],int num_words)
{
    int i,j;

        for(i=0 ; i<num_words ; i++)
                if(put_word(Table, words[i])==-1)
                        return 0;

        for(i=0 ; i < SIZE ; i++)
                for(j=0 ; j<SIZE ; j++)
                        if(Table[i][j]=='\0')
                         Table[i][j] = rand()%(26)+97;
        return 1;
}


int find_word(char Table[SIZE][SIZE], char word[],int x,int y)
{
      int i;
      int counter=0;
      int x1,y1;

          x1=x;
      y1=y;

          for(i=0 ; i<strlen(word) || Table[x1][y1]!=word[i] ; y1++,i++)
             if((Table[x1][y1]==word[i]) || (Table[x1][y1] == word[i]-32))
                        counter++;
          if(counter==strlen(word))
              return 0;

       x1=x;
       y1=y;
       counter=0;

      for(i=0 ; i<strlen(word)||Table[x1][y1]!=word[i] ; x1++,i++)
            if((Table[x1][y1]==word[i]) || (Table[x1][y1] == word[i]-32))
                   counter++;
          if(counter==strlen(word))
                   return 1;

          x1=x;
          y1=y;
          counter=0;

      for(i=0 ; i<strlen(word)||Table[x1][y1]!=word[i] ; y1++,x1++,i++)
            if((Table[x1][y1]==word[i]) || (Table[x1][y1] == word[i]-32))
                    counter++;
          if(counter==strlen(word))
                   return 2;

          x1=x;
          y1=y;
      counter=0;

      for(i=0;i<strlen(word)||Table[x1][y1]!=word[i];y1--,x1++,i++)
            if((Table[x1][y1]==word[i]) || (Table[x1][y1] == word[i]-32))
                    counter++;
          if(counter==strlen(word))
              return 3;
```

```
    return -1;
  }

void highlight_solution(char Table[SIZE][SIZE], char *word,int x, int y, int
direction)
{
        int i;
        switch(direction)
        {
            case 0:
                      for(i=0 ; i<strlen(word) ; y++,i++)
                                     Table[x][y] = word[i] - 32;
                         break;
            case 1:
                         for(i=0;i<strlen(word);x++,i++)
                          Table[x][y] = word[i] - 32;
                      break;
            case 2:
                         for(i=0;i<strlen(word);y++,x++,i++)
                              Table[x][y] = word[i] - 32;
                  break;
            case 3:
                         for(i=0;i<strlen(word);y--,x++,i++)
                              Table[x][y] = word[i] - 32;
                  break;
    }
}

int get_all_locations(char Table[SIZE][SIZE],char words[WORDS_NUM][LENGTH],int
num_words)
{
     int i;
         int success=0,failure=0;
         int choice,x,y,direction;
         int flags[WORDS_NUM];

         for (i=0 ; i<WORDS_NUM ; i++)
                 flags[i]=0;

     while(success < num_words)
         {
                        show_table(Table,words,flags,num_words);

                        do{
                                printf("Chose word number:");
                                scanf("%d",&choice);
                                if(choice<1 || choice>num_words)
                                        printf("Chose right option!\n");
                        }while(choice<1 || choice>num_words);


                        if(flags[choice-1]==0)
                        {
                                printf("\nEnter location (row and column) of the
word:");
                                printf("%s",words[choice-1]);
                                printf("\nTo end the game please enter non positive
values\n");
                                printf("Enter row number:");
                                scanf("%d",&x);
                                printf("Enter column number:");
```

```
                                scanf("%d",&y);

                                if(x<=0 || y<=0)
                                {
                                   printf("*end of game*.\n");
                                   return 0;
                                }

                                if((direction = find_word(Table,words[choice-1],x-1,y-
1)) != -1)
                                {
                                   printf("**Right answer**\n");
                                   flags[choice-1]=1;
                                   highlight_solution(Table,words[choice-1],x-1,y-
1,direction);
                                   success++;
                                }

                                else
                                {
                                        printf("##Wrong answer##\n");
                                        failure++;
                                }
                        }

                        else
                           printf("you have already guessed this word\n");
     }

     if(success == num_words)
         {
            show_table(Table,words,flags,num_words);
            printf("\nYou have guessed all the words!\n");
            printf("The final score is: %d/%d\n",success,success+failure);
         }

         return 1;
}
```

```
/* main */

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include "tifzoret.h"

int main()
{
        char Table[SIZE][SIZE];
        char Words[WORDS_NUM][LENGTH];
        int num_words=0;

        srand(time(0));
        init_table(Table);

        if (assign_words(Table,Words,num_words=get_words(Words)) == 0)
        {
                printf("Can't put the words\n");
                return 1;
        }

        get_all_locations(Table,Words,num_words);

        return 0;
}
```

מבוא למדעי המחשב
סמסטר ב' תשס"ה

תרגיל מס' 11
תאריך הגשה: 19/6/05

.1
פרמוטציה של סדרת מספרים היא סידור כלשהו של כל אברי הסדרה. למשל, 3,0,6,5,2 היא פרמוטציה של הסדרה 5,6,0,2,3. הגדירו פונקציה המקבלת מערך של שלמים ואת אורכו, n, ומחזירה פרמוטציה **אקראית** של אברי המערך. המשמעות של פרמוטציה אקראית היא שבכל הפעלה של הפונקציה תתקבל פרמוטציה כלשהי של סדרת הקלט, כך שההסתברות לקבלת כל פרמוטציה היא שווה. ניתן להשתמש בפונקציה toss המקבלת מספר שלם k ומחזירה מספר שלם אקראי בין 0 ל-k-1. אין להשתמש באף פונקציה אחרת. סיבוכיות זמן: O(n). סיבוכיות מקום: O(1) (כלומר, על הפונקציה לסדר מחדש את אברי המערך שהיא מקבלת ללא שימוש במערך נוסף).
את הפונקציה toss עליכם להגדיר.

.2
הגדירו פונקציה המקבלת מערך של שלמים, a, ואת גודלו n, ומספר שלם max. על הפונקציה להדפיס מספר גדול ככל האפשר של אברי המערך, ובלבד שסכום המספרים המודפסים לא יעלה על max.
למשל, אם הקלט הוא הסדרה 6,1,-8,-12,37,38,39,15 ו-max הוא 0, על הפונקציה להדפיס את המספרים 6,1,-8,-12. סדר ההדפסה אינו משנה.

על הפונקציה לעבוד בזמן O(N log N). פתרונות בסיבוכיות גבוהה יותר לא יתקבלו. **אין הגבלה על סיבוכיות מקום**.

```
void print_max(int a[], int n, int max)
```

.3
מייצגים קבוצות של מספרים שלמים באמצעות מערכים המכילים את אברי הקבוצות (ללא חזרות). הגדירו פונקציה בשם intersect המקבלת שני מערכים של שלמים, a ו-b, ואת אורכיהם lena ו-lenb, ומחזירה מערך c (עם אורכו lenc) שבו אברי ה**חיתוך** של הקבוצות המיוצגות על ידי a ו-b (שוב, ללא חזרות).
למשל, אם a הוא הקבוצה {38,39,6,51,1,-8} ו-b הקבוצה {6,1,-8,-12,37}, על הפונקציה להחזיר מערך שייצג את הקבוצה {6,1,-8}.

על הפונקציה לעבוד בזמן O(N log N), כש-N הוא הגדול מבין lena ו-lenb. פתרונות בסיבוכיות גבוהה יותר לא יתקבלו.

```
void intersect(int a[], int lena, int b[], int lenb,  int c[], int *lenc)
```

.4
מטריצה A נקראת <u>ממוינת</u> אם כל שורה וכל עמודה בה ממוינת (שורות ממוינות משמאל לימין ועמודות ממוינות מלמעלה למטה).
כתבו פונקציה

```
int findNum (int matrix[SIZE][SIZE], int n, int number)
```

המקבלת מטריצה <u>ממוינת</u> (matrix) בגודל $n \times n$ ומספר (number) ומחזירה 1 באם המספר נמצא במטריצה, 0 אחרת.
ניתן להניח כי גודל המערך הדו-מימדי SIZE מוגדר בעזרת define וכן כי גודל המטריצה n מקיים n<SIZE.
סיבוכיות זמן: O(n)
סיבוכיות מקום: O(1)

/* permutation.c */

```
#include <stdio.h>


int toss(int k) {

        return random()%k;
}

void permutation(int arr[],int n) {

        int i,tmp,index;

        for(i=0 ; i<n ;i++) {

                index=toss(n-i)+i;

                tmp=arr[i];
                arr[i]=arr[index];
                arr[index]=tmp;
        }
}
```

/* max.c */

```
#include <stdio.h>

void sort (int array[], int size); /* mergesort - O(n log n) */

void print_max(int a[], int n, int max)
{
  int sum=0,i;

  sort(a,n);
  for (i=0; i<n && a[i]<0; i++) {
          sum += a[i];
  }
  if (sum<=max) {
          for (j=0; j<i; j++) {
                  printf("%d ", a[j]);
          }
  }
  while (i<n && (sum+=a[i])<=max) {
    printf("%d ",a[i++]);
  }
  printf("\n");
}
```

/* intersect.c */

```
#include <stdio.h>

void sort (int array[], int size);
int binsearch (int n, int v[], int low, int high);


void intersect(int a[], int lena, int b[], int lenb, int c[], int *lenc)
{
        int i=0;

        *lenc=0;
        sort(b,lenb);
        for(i=0 ; i<lena ; i++)
                if(binsearch( a[i],b,0,lenb-1) != -1 ) {
                        c[*lenc]=a[i];
                        (*lenc)++;
                }
}
```

/* matrix.c */

```
#include <stdio.h>
#define SIZE 20

int findNum (int matrix[SIZE][SIZE], int n, int number)
{
        int i=0, j=n-1;

        if(n==0)
                return 0;

        while(matrix[i][j] != number) {
                if(matrix[i][j] > number)
                        j--;
                else
                        i++;
                if(i==n || j== -1)
                        return 0;
        }
        return 1;
}
```

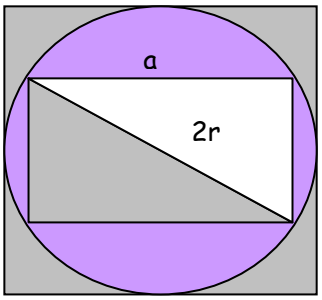
מבוא למדעי המחשב

סמסטר א' תשס"ה

תרגיל מס' 2
תאריך הגשה: 8/11/04

תרגיל

נתון השרטוט הבא, המתאר מלבן חסום במעגל שחסום בריבוע:



רדיוס המעגל הוא r, ולכן האלכסון של המלבן החסום וצלע הריבוע החוסם הם באורך 2r. הצלע הארוכה של המלבן היא באורך a.

כתבו תוכנית ששמה calc.c, אשר תקלוט מהמשתמש ערכים עבור r ו-a, ותחשב את:

1. שטח המלבן.
2. שטח העיגול.
3. שטח הריבוע החוסם את המעגל.
4. שטח המשולש הצבוע באפור.

לבסוף תדפיס התוכנית את תוצאות החישובים.

הוראות עזר

על מנת לחשב שורש של מספר מסויים, יש להשתמש בפונקציה sqrt() המוגדרת בספרייה math.h . הפונקציה sqrt מקבלת פרמטר מטיפוס double ומחזירה את שורשו.

לדוגמא:

```
/* sqrt example */
#include <stdio.h>
#include <math.h>

int main ()
{
  double param, result;
  param = 1024.0;
  result = sqrt (param);
  printf ("sqrt(%lf) = %lf\n", param, result );
  return 0;
}
```

שימו לב כי יש לכלול את הספרייה math.h בתחילת הקוד על מנת להשתמש בפונקציה הנ"ל.

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

#define PIE 3.14159

int main()
{

        double radius, a;
        double area_rect;

        printf("please enter radius value:"); /* get the values from the user */
        scanf("%lf",&radius);
        printf("please enter 'a' value:");
        scanf("%lf",&a);

        area_rect = sqrt(4*radius*radius - a*a) * a;  /* print the results */
        printf("The area of the rectangle is: %f\n", area_rect);
        printf("The area of the circle is: %f\n", PIE*radius*radius);
        printf("The area of the square is: %f\n", 4*radius*radius);
        printf("The area of the triangle is: %f\n", area_rect/2);

        return 0;

}
```

מבוא למדעי המחשב

סמסטר א' תשס"ה

תרגיל מס' 3

תאריך הגשה: 15/11/04

<u>תרגיל 1</u>

כתבו תכנית המבצעת פעולות שונות על מספרים שלמים וחיוביים בני 4 ספרות בלבד!
כמפורט להלן.
התוכנית תקלוט מספר בן 4 ספרות, ותציג את תפריט הפעולות הבא למשתמש :

(1) - חשב את סכום הספרות של המספר. (עבור 1987 , התוצאה תהיה 25)
(2) - הדפס את המספר עם סדר ספרות הפוך (עבור 1987 התוצאה תהיה 7891 ,
ועבור 2000 התוצאה תהיה 2) .
(3) - יציאה (מבלי לבצע אף חישוב).

בשלב זה, יש לבקש מהמשתמש לבחור באחת מהאפשרויות, לבצע את החישוב הנדרש, ולהדפיס את תוצאת החישוב.
<u>הערה</u>: יש לוודא כי מספר הקלט הוא בן 4 ספרות. אחרת, התוכנית תדפיס הודעת שגיאה למשתמש.

<u>תרגיל 2</u>

כתבו תכנית אשר מקבלת כקלט תו כלשהו, ומדפיסה כפלט את הסיווג של תו הקלט עפ"י הקטגוריות הבאות:

- ספרה (הספרות 0 עד 9).
- תנועה (האותיות A, E, I, O, U, a, e, i, o, u).
- עיצור (אותיות האלף-בית האנגלי, אשר אינן תנועות).
- רווח.
- תו אחר.

למשל:
עבור התו a יודפס "vowel" .
עבור התו 0 יודפס " digit" .
עבור התו יודפס " space".
ועבור התו & יודפס " תו כלשהו שאינו אחד מהקטגוריות המפורטות".

לנוחיותכם, ניתן להשתמש במספר פונקציות שימושיות, אשר מוגדרות ב-ctype.h המקבלות תו ומחזירות ערך שונה מאפס או אפס לפי תפקידן:

- int isalpha(int c); - תחזיר ערך שונה מאפס אם התו הוא אות באב"ב, ואפס אחרת.
- int isdigit(int c); - תחזיר ערך שונה מאפס אם התו הוא סיפרה, ואפס אחרת.
- int isspace(int c); - תחזיר ערך שונה מאפס אם התו הוא רווח, ואפס אחרת.
- int ispunct(int c); - תחזיר ערך שונה מאפס אם התו אינו אחד מהבאים אות באב"ב, ספרה, בקרה, או רווח , ואפס אחרת

השימוש בפונקציות אלה אינו חובה, אך הוא יקל עליכם במימוש במידה ניכרת.

```c
/* numbers.c */
#include <stdio.h>

int main()
{

        int number,option;
        int dig_1, dig_10, dig_100, dig_1000;

        printf("Please enter a number:");
        scanf("%d",&number);

        if(number < 1000 || number > 9999) {
                printf("wrong input!\n");
                return 1;
        }

        printf("Please choose one of the following options:\n");
        printf("(1) count the sum of the number's digits.\n");
        printf("(2) print the number in reverse order.\n");
        printf("(3) Exit.\n");
        scanf("%d",&option);

        switch(option)
        {
                case 1:
                        dig_1= number%10;             /* Extracting the 1th digit */
                        dig_10= (number /10) %10;     /* Extracting the 10th digit */
                        dig_100= (number /100)%10;    /* Extracting the 100th digit */
                        dig_1000= (number /1000)%10;  /* Extracting the 1000th digit*/
                        printf("The sum of digits of the number %d is: %d",number,
dig_1 + dig_10 + dig_100 + dig_1000);
                        break;

                case 2:
                        dig_1= number%10;
                        dig_10= (number /10) %10;
                        dig_100= (number /100)%10;
                        dig_1000= (number /1000)%10;

                        printf("The reverse number of %d is: ",number);
                        if( dig_1 != 0 )
                                printf("%d",dig_1);
                        if( (dig_1 != 0) || (dig_10 != 0) )
                                printf("%d",dig_10);
                        if( (dig_1 != 0) || (dig_10 != 0) || (dig_100 != 0) )
                                printf("%d",dig_100);
                        printf("%d\n",dig_1000);
                        break;

                case 3:
                        printf("Exit\n");
                        break;

                default:
                        printf("You didn't choose a valid option.\n");
        }

        return 0;
}
```

```
/* characters.c */

#include <stdio.h>
#include <ctype.h>

int main()
{
        char ch;
        printf("Enter a character:");
        scanf("%c",&ch);


        if (isalpha(ch)) {
                switch (ch) {
                case 'A' :
                case 'E' :
                case 'I' :
                case 'O' :
                case 'U' :
                case 'a' :
                case 'e' :
                case 'i' :
                case 'o' :
                case 'u' :
                        printf("The character %c is a vowel.\n",ch);
                        break;
                default :
                        printf("The character %c is a consonant.\n",ch);
                        break;
                }

        } else if (isdigit(ch))
                printf("The character %c is a digit.\n",ch);
        else if (isspace(ch))
                printf("The character %c is a space.\n",ch);
        else if (ispunct(ch))
                printf("The character %c is not a digit and not an alphabethical
character.\n",ch);
        else {
                printf("Unexpected character |%c| in input\n",ch);
                return(1);
        }
        return 0;
}
```

מבוא למדעי המחשב

סמסטר א' תשס"ה

תרגיל מס' 5
תאריך הגשה: 6/12/04

Nim

עליכם לממש משחק ידוע בשם Nim המנוהל בין שני שחקנים. לוח המשחק נראה כך:

|	|	|	|	|	|
|	|	|	|		

ישנן שתי ערימות המכילות מספר גפרורים מסוים. כל שחקן בתורו יכול לשלוף מספר כלשהו של גפרורים מערימה אחת בלבד. על השחקן לבחור בתורו את מספר הערימה ממנה הוא מעוניין להסיר גפרורים, ואת מספר הגפרורים להסרה. מי שמסיר את הגפרור האחרון הוא המנצח.
עליכם לממש משחק מחשב המתנהל בין המשתמש למחשב, בו תמיד המחשב מנצח! יש לאפשר למחשב לבחור האם לשחק ראשון או לא.
לאחר סיום כל משחק, התוכנית תדפיס מי ניצח, ותאפשר למשתמש לשחק שוב, עד שהוא יבחר לפרוש מהמשחק.

יש לממש את הפונקציות הבאות:

initialize_game – הפונקציה תקלוט מהמשתמש את מספר הגפרורים ההתחלתי בכל ערימה.
display_game – הפונקציה תדפיס את המצב הנוכחי של המשחק .
לדוגמא: בערימה 1 יש 3 גפרורים ובערימה 2 יש 4 גפרורים -

|	|	|		
|	|	|	|	

game_over – הפונקציה תבדוק אם נותרו גפרורים בערימות. הפונקציה תחזיר ערך true אם אכן נותרו גפרורים ו- false אם לא.
user_play – הפונקציה תבקש מהמשתמש לבחור את מספר הגפרורים להסרה ואת מספר הערימה הרצויה. לאחר-מכן, הפונקציה תעדכן את המספר הנוכחי של הגפרורים בערימות.
הערה: יש לקלוט מה- user מספר חוקי של ערימה ושל גפרורים להסרה.
computer_play – הפונקציה תנהל את אסטרטגיית המשחק של המחשב. הפונקציה תבחר את מספר הגפרורים הרצוי להסרה וכן את מספר הערימה המבוקשת, כך שהמחשב תמיד ינצח בסופו של דבר במשחק. לאחר-מכן, הפונקציה תעדכן את המספר הנוכחי של הגפרורים בערימות.
manage_game – הפונקציה תבצע אתחול של המשחק ותנהל את המשחק בין שני השחקנים, כל אחד בתורו. לאחר ביצוע כל תור, הפונקציה תדפיס את המצב הנוכחי של המשחק, עד סופו.
לבסוף הפונקציה תדפיס מי המנצח במשחק.

פונקציית ה- **main** תפעיל את המשחק ע"י קריאה לפונקציה manage_game . לאחר סיום המשחק, יש לשאול את המשתמש האם ברצונו לשחק פעם נוספת. אם כן, המשחק ינוהל שוב, אחרת התוכנית תסיים את פעולתה. התוכנית תמשיך את פעולתה עד שהמשתמש יבקש לסיים את המשחק.

```
#include <stdio.h>


void play_game();
void initialize_piles();
void display_piles();
int all_done();
int get_human_move();
int select_computer_move();

int pile1, pile2;


/*
        This function decides who start the game andmanages the game between the user
and the
        computer step by step. The function display the current situation after every
move.
        At the end, the function prints who won the game   */


void play_game()
{
    int chips_removed;
        int turn;

        printf("Initial Situation:\n");
    display_piles();

        if(pile1==pile2) { /* user start playing */
                printf("User starts playing.\n");
                turn= 0;
        }
        else {
                printf("Computer starts playing.\n");
                turn= 1;   /* computer start playing */
        }

    while (1) {

        if (turn) {
            chips_removed = select_computer_move();
            printf("The computer has removed %d \n",chips_removed);

            if (all_done()) {
                printf("*** The computer has won! ***\n");
                break;
            }
        }
        else {
            get_human_move();
            if (all_done()) {
                printf("*** Congratulations - you have won! ***\n");
                break;
            }
        }

        /* display the new situation */
        display_piles();

        /* it's now the other player's turn */
```

```
        turn = !turn;
    }
}

/*  This function gets the initial state from the user */

void initialize_piles()
{
    printf("Please enter number of matches in pile 1:");
        scanf("%d",&pile1);
    printf("Please enter number of matches in pile 2:");
        scanf("%d",&pile2);
}



/* This function displays the number of chips in each of the piles. */

void display_piles()
{
     int i;
     printf("%d  :  %d  \n",pile1,pile2);
     for (i=0 ; i<pile1*2 ; i++)
               printf("-");
     printf("\n");

     for (i=0 ; i<pile1 ; i++)
               printf("| ");
     printf("\n");

     for (i=0 ; i<pile1*2 ; i++)
               printf("-");
     printf("\n");
     for (i=0 ; i<pile2 ; i++)
               printf("| ");
     printf("\n");

     for (i=0 ; i<pile2*2 ; i++)
               printf("-");
     printf("\n");
}



/* returns 'true' if there are no more chips left in the piles */


int all_done()
{
    if ((pile1 > 0) || (pile2 > 0)) {
         return 0;   /* there's something in pile 'n', so we're not done */
        }


    return 1; /* there are no chips in any of the piles, so we're done */
}
```

```
/* obtains a valid human move by repeatedly asking for a move until valid
   input is supplied.  assumes that there is at least one legal move. */


int get_human_move()
{
        int pile, chips;
    while (1) {
        printf("Please enter pile number and chips to remove: ");
        scanf("%d%d",&pile,&chips);

        if ( ((pile == 1) && (chips <= pile1)) || ((pile == 2) && (chips <= pile2)) )
{
            break;                      /* we have a valid move */
        }

        printf("**** Invalid pile number or number of chips. ****\n");
    }
        if(pile == 1)
                pile1 -= chips;
        if(pile == 2)
                pile2 -= chips;

        return chips;
}



/* given the current suituation, selects a computer move.
   it is assumed that there is at least one possible move. */


int select_computer_move()
{
    int extrabits = 0;
    int mypile, mysticks;

        extrabits = extrabits ^ pile1;
        extrabits = extrabits ^ pile2
                ;
    if (extrabits == 0) {

            if (pile1) {
                        mypile = 1;
                        mysticks = 1;
            }
                if (pile2) {
                        mypile = 2;
                        mysticks = 1;
            }
    } else {

            if ((pile1 ^ extrabits) < pile1) {
                        mypile = 1;
                        mysticks = pile1 - (pile1 ^ extrabits);
            }
                if ((pile2 ^ extrabits) < pile2) {
                        mypile = 2;
                        mysticks = pile2 - (pile2 ^ extrabits);
            }
```

```
    }

        if(mypile ==1)
                pile1 -= mysticks;
        if(mypile ==2)
                pile2 -= mysticks;
        return mysticks;
}



int main()
{
        int play=1;
        while(play) {

                initialize_piles();
                play_game ();

                printf("Do you want to play again? (1/0)");
                scanf("%d",&play);
        }

        return 0;
}
```

מבוא למדעי המחשב
סמסטר א' תשס"ה

תרגיל מס' 7
תאריך הגשה: 28/12/04

נייצג **תמונה בשחור לבן** ע"י מערך דו-מימדי NxM, שכל תא בו מכיל 1 עבור נקודה שחורה, ו-0 עבור נקודה לבנה. כאשר N ו- M מוגדרים ב- define# ל- 20.

בקלט שלכם יכולה להופיע תמונה של הספרה '1' או של הספרה '0' .

לדוגמא:

```
00000000000000000000
00000000000000000000
00001111111111100000
00010000000000010000
00100000000000001000
00100000000000001000
00100000000000001000
00100000000000001000
00100000000000001000
00100000000000001000
00100000000000001000
00100000000000001000
00100000000000001000
00100000000000001000
00100000000000001000
00100000000000001000
00100000000000001000
00100000000000001000
00010000000000010000
00001111111111100000
00000000000000000000
00000000000000000000
```

```
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000001100000000
00000000010100000000
00000000100100000000
00000000000100000000
00000000000100000000
00000000000100000000
00000000000100000000
00000000000100000000
00000000000100000000
00000000000100000000
00000000000100000000
00000000000100000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
```

ראו דוגמאות קלט נוספות בקישור הבא:

http://cs.haifa.ac.il/courses/csintro/assignments/pic_input.txt

עליכם לכתוב תוכנית המקבלת כקלט תמונה בשחור לבן ומדפיסה את הספרה המופיעה בתמונה (0 או 1).
יש לממש אלגוריתם המתאים לתמונות המופיעות בדוגמאות הקלט ולתמונות אחרות הדומות בעיקרן לדוגמאות הקלט.
השתדלו לכתוב אלגוריתם כוללני ככל האפשר.

יש לחלק את התוכנית לפונקציות באופן ברור והגיוני.

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

#define N 20
#define M 20

void sanity_check_picture(int p[N][M]);
void read_picture(int p[N][M]);
void print_picture(int p[N][M]);
int check_image(int p[N][M]);

void sanity_check_picture(int p[N][M]) {
        int i,j;
        for(i=0;i<N;++i) {
                for(j=0;j<M;++j) {
                        if (!(0==p[i][j] || 1==p[i][j])) {
                                printf("Sanity check detected an unexpected value "
                                        "in row %d and column %d: %c\n", i,j,p[i][j]);
                                exit(EXIT_FAILURE);
                        }
                }
        }
}


void read_picture(int p[N][M]) {
        int i=0,j=0,c;
        while( EOF!=(c=getchar())) {
                switch(c) {
                        case ' ':
                        case '\t':
                                break;  /* ignore spaces and tabs */
                        case '\n':
                                ++i;
                                j=0;
                                break; /* newline character means a line is over */
                        case '0':
                        case '1':
                                p[i][j++]=c-'0'; /* convert '1' or '0' to 1 or 0 */
                                break;
                        default:
                                printf("Invalid input in row %d and column %d: %c\n",
                                                i,j,c);
                                exit(EXIT_FAILURE);
                }
        }
        sanity_check_picture(p); /* make sure all cells are filled correctly */
}

void print_picture(int p[N][M]) {
        int i,j;
        for(i=0;i<N;++i){
                for(j=0;j<M;++j) {
                        printf("%d ", p[i][j]);
                }
                putchar('\n');
        }
}
```

```
int check_image(int p[N][M])
{
        int i=0,j=0;
        int tmp_i=-1, tmp_j=-1, last_i=-1, last_j=-1;
        int flag=0;

        for(i=0 ; i<N ; i++) {
                for(j=0 ; j<M ; j++) {
                        if(p[i][j]==1) {
                                tmp_i=i;
                                tmp_j=j;
                                break;
                        }
                }
                if(p[i][j]==1)
                                break;
        }


        do{

                if(p[tmp_i-1][tmp_j] && (tmp_i-1 != last_i || tmp_j != last_j)) {
/* direction: up */
                        last_i=tmp_i;
                        last_j=tmp_j;
                        tmp_i=tmp_i-1;
                }

                else if(p[tmp_i+1][tmp_j] && (tmp_i+1 != last_i || tmp_j != last_j))
{  /* direction: down  */
                        last_i=tmp_i;
                        last_j=tmp_j;
                        tmp_i=tmp_i+1;
                }

                else if(p[tmp_i][tmp_j-1] && (tmp_i != last_i || tmp_j-1 != last_j))
{  /* direction: left */
                        last_i=tmp_i;
                        last_j=tmp_j;
                        tmp_j=tmp_j-1;
                }

                else if(p[tmp_i][tmp_j+1] && (tmp_i != last_i || tmp_j+1 != last_j))
{  /* direction: right */
                        last_i=tmp_i;
                        last_j=tmp_j;
                        tmp_j=tmp_j+1;
                }

                else if(p[tmp_i-1][tmp_j-1] && (tmp_i-1 != last_i || tmp_j-1 !=
last_j)) { /* direction: up left */
                        last_i=tmp_i;
                        last_j=tmp_j;
                        tmp_i=tmp_i-1;
                        tmp_j=tmp_j-1;

                }

                else if(p[tmp_i-1][tmp_j+1] && (tmp_i-1 != last_i || tmp_j+1 !=
last_j)) { /* direction: up right */
                        last_i=tmp_i;
```

```
                        last_j=tmp_j;
                        tmp_i=tmp_i-1;
                        tmp_j=tmp_j+1;
                }


                else if(p[tmp_i+1][tmp_j-1] && (tmp_i+1 != last_i || tmp_j-1 !=
last_j)) {  /* direction: down left */
                        last_i=tmp_i;
                        last_j=tmp_j;
                        tmp_i=tmp_i+1;
                        tmp_j=tmp_j-1;
                }


                else if(p[tmp_i+1][tmp_j+1] && (tmp_i+1 != last_i || tmp_j+1 !=
last_j)) {  /* direction: down right */
                        last_i=tmp_i;
                        last_j=tmp_j;
                        tmp_i=tmp_i+1;
                        tmp_j=tmp_j+1;
                }
                else
                        return 1;

        }while(tmp_i != i || tmp_j != j);

        return 0;
}


int main()

{
        int i,j;
        int p[N][M];

        read_picture(p);
        printf("The image is: %d",check_image(p));

        return 0;

}
```

מבוא למדעי המחשב
סמסטר א' תשס"ה

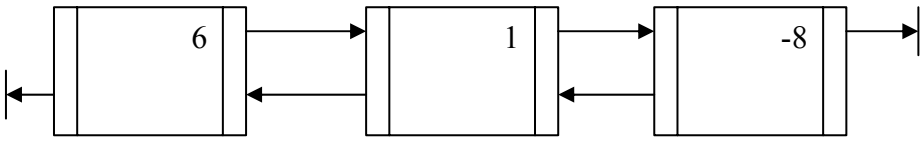
תרגיל מס' 8
תאריך הגשה: 5/1/05

.1

רשימה מקושרת דו כיוונית היא רשימה מקושרת אשר כל צומת בה מקושר, דרך מצביע, הן לאבר שאחריו והן לזה שלפניו (אם הם קיימים). מממשים רשימה כזו על ידי תאים מהטיפוס:

```
typedef struct cell *CellPtr;
typedef struct cell {
     int contents;
     CellPtr next;
     CellPtr prev;
} Cell;
```

באופן גראפי, רשימה המכילה את הנתונים 6,1,-8 תראה כך:



א. הגדירו פונקציה בשם print_list המקבלת מצביע ל**תחילת** רשימה דו כיוונית ומדפיסה את אברי הרשימה.

void print_list (CellPtr list)

ב. הגדירו פונקציה בשם print_rev המקבלת מצביע ל**סוף** רשימה דו כיוונית ומדפיסה את אברי הרשימה מהסוף להתחלה.

void print_rev (CellPtr list)

ג. הגדירו פונקציה בשם insert_first המקבלת מצביע לתחילת רשימה דו כיוונית ומספר שלם input ומוסיפה אבר שמכיל את input לתחילת הרשימה. הפונקציה צריכה להחזיר מצביע לתחילת הרשימה המעודכנת.

CellPtr insert_first (CellPtr list, int input)

ד. הגדירו פונקציה בשם remove_first המקבלת מצביע לתחילת רשימה דו כיוונית ומצביע ל-int בשם output. על הפונקציה להסיר את האבר הראשון ברשימה, לשחרר את המקום שהוקצה עבורו ולהחזיר מצביע לתחילת הרשימה המעודכנת. כמו כן עליה להחזיר, דרך המצביע output, את תוכן האבר שהוסר מהרשימה.

CellPtr remove_first (CellPtr list, int *output)

כתבו פונקציית Main על מנת לבדוק את פעולת הפונקציות שהגדרתם.

2. שאלה זו עוסקת ברשימות מקושרות שכל תא בהן מוגדר על ידי הטיפוס:

```
typedef struct cell *CellPtr;
typedef struct cell {
     int contents;
     CellPtr next;
} Cell;
```

הגדירו פונקציה בשם copy_list המקבלת מצביע לתחילת רשימה ומחזירה מצביע לתחילת רשימה שהיא העתק של רשימת הקלט (כלומר, כל תא בה הוא העתק של התא המקביל ברשימת הקלט). אין לפגוע ברשימת הקלט.

CellPtr copy_list (CellPtr list)

כתבו פונקציית Main על מנת לבדוק את פעולת הפונקציה שהגדרתם.

```c
/* doubly_linked_list.c */
#include <stdio.h>
#include <stdlib.h>

typedef struct cell *CellPtr;
typedef struct cell {
        int contents;
        CellPtr next;
        CellPtr prev;
} Cell;

void print_list(CellPtr list);
void print_rev(CellPtr list);
CellPtr insert_first (CellPtr list, int input);
CellPtr remove_first (CellPtr list, int* input);


void print_list(CellPtr list)
{
    CellPtr tmp;
    for (tmp=list ; tmp != NULL ; tmp=tmp->next)
        printf("%d  ",tmp -> contents);
}

void print_rev(CellPtr list)
{
    CellPtr tmp;
    for (tmp=list ; tmp != NULL ; tmp=tmp->prev)
        printf("%d  ",tmp -> contents);
}


CellPtr insert_first (CellPtr list, int input)
{
        CellPtr tmp;

        if (!(tmp=(CellPtr)malloc(sizeof(struct cell))))
           exit(EXIT_FAILURE);
        tmp -> contents = input;
        if (list == NULL) {    /* if the list is empty */
           tmp -> next = NULL;
           tmp -> prev = NULL;
           list = tmp;
           return list;
        }
        tmp -> next = list;
        list -> prev = tmp;
        tmp -> prev = NULL;
        list = tmp;
        return list;
}

CellPtr remove_first (CellPtr list, int* input)
{
        CellPtr tmp;

        if (list==NULL)    /* if the list is empty */
           return NULL;

        if (list->next == NULL) {  /* if the list contains one element */
           *input = list->contents;
```

```
            free(list);
            return NULL;
        }
        tmp = list->next;
        tmp -> prev = NULL;
        *input = list->contents;
        free(list);
        return tmp;
}

int main()
{

        CellPtr l1,l2,l3,list,last;
        int value;

        l1=(CellPtr)malloc(sizeof (struct cell));
        l2=(CellPtr)malloc(sizeof (struct cell));
        l3=(CellPtr)malloc(sizeof (struct cell));

        l1->contents = 6;
        l2->contents = 1;
        l3->contents =-8;

        l1->next=l2;
        l2->next=l3;
        l3->next=NULL;

        l3->prev=l2;
        l2->prev=l1;
        l1->prev=NULL;

        list=l1;
        last=l3;

        printf("\n");
        print_list(list);

        printf("\n");
        print_rev(last);

        printf("\n");
        list=insert_first(list,23);

        printf("\n");
        print_list(list);

        printf("\n");
        print_rev(last);

        list = remove_first(list,&value);
        printf("\n");
        print_list(list);
        printf("\nThe deleted value is %d\n",value);

        free(l1);
        free(l2);
        free(l3);
        free(list);
        free(last);
}
```

```
/* copy_list.c */

#include <stdio.h>

typedef struct cell *CellPtr;
typedef struct cell {
        int contents;
        CellPtr next;
} Cell;

CellPtr copy_list (CellPtr list);
void print_list(CellPtr tmp);

CellPtr copy_list (CellPtr list)
{
        CellPtr head,tmp,new_cell;
        tmp=NULL;

        if(list==NULL) {
                printf("The list is empty.\n");
                return NULL;
        }
        else if(NULL==(new_cell=(CellPtr)malloc(sizeof(Cell))))
                exit(1);
        new_cell -> contents = list -> contents;
        tmp=new_cell;
        head=new_cell;
        list=list->next;

        while(list != NULL) {

                if(NULL==(new_cell=(CellPtr)malloc(sizeof(Cell))))
                        exit(1);
                new_cell -> contents = list -> contents;
                tmp -> next = new_cell;
                tmp=tmp->next;
                tmp->next=NULL;
                list=list->next;
        }
        tmp->next=NULL;

        return head;
}


void print_list(CellPtr tmp)
{
        while(tmp != NULL)
        {
                printf("%d ",tmp->contents);
                tmp=tmp->next;
        }
        putchar('\n');
}
```

```
int main()
{
        CellPtr l[4],tmp1,tmp2;
        int i;

        for(i=0;i<4;i++)
                l[i]=(CellPtr)malloc(sizeof(Cell));
        for(i=0;i<4;i++)
                l[i]->contents=i+1;
        for(i=0;i<4;i++)
                l[i]->next =l[i+1];
        l[3]->next=NULL;

        tmp1=l[0];
        printf("List 1: ");
        print_list(tmp1);

        tmp2=copy_list(l[0]);
        printf("copy List : ");
        print_list(tmp2);

        return 0;
}
```

Department of Computer Science
University of Haifa

החוג למדעי המחשב
אוניברסיטת חיפה

Mount Carmel, 31905 Haifa, Israel Phone: +972-4-8240259 Fax: +972-4-8249331 הר הכרמל, חיפה 31905 טל. 04-8240259 פקס: 04-8249331

Introduction to Computer Science
Homework assignment 1

Submission deadline: 30/10/2002 at midnight

October 21, 2002

1 The problem

Write a computer program that reads five numbers and prints the sum of the five numbers.

2 Submission guidelines

1. Your program should comply to ANSI-C, otherwise your grade will be zero.
2. Your program should be submitted on-line through the course site before the deadline. Late submissions or no submissions will result in a grade of zero.
3. Submit only your source code (the .c file), do not submit any other file.
4. You should not submit any paper prinout of your source code.
5. Think of problems you might have with the input or with your calculations and implement a solution (e.g. what if the user gives you incompatible input?)

3 Submission process

1. First go to http://cs.haifa.ac.il/courses/csintro/course_registration_form.html and fill out the form in order to obtain a username and a password. Note that you don't get your username and password immediately, so don't postpone this part to the last minute. Getting your username and passwrd is something you do only one time, so if you already did it, don't do it again!
2. Next you need to have your source code file ready on your local computer.
3. Next you need to fill out the upload form at http://cs.haifa.ac.il/courses/csintro/upload_form.html, which means putting your username and password, specifying the homework number (in this case the homework number is 1), and finally selecting the file to upload. Then you click on the SUBMIT button.
4. Make sure the operation was successful!

```
/*
*
* Introduction to Computer Science
* Winter 2002-2003
* Suggested solution for homework assignment #1
*
* The program should read 5 numbers from the standard input and print
* to the standard output the sum of the five numbers.
* You may assume legal input and you don't have to worry about overflow
* and other problems, and you don't have to do any error handling.
*
*/
#include <stdio.h>

int main() {
        float num1,num2,num3,num4,num5,sum;
        scanf("%f%f%f%f%f",&num1,&num2,&num3,&num4,&num5);
        sum = num1+num2+num3+num4+num5;
        printf("%f+%f+%f+%f+%f=%f\n",num1,num2,num3,num4,num5,sum);
        return 0;
}
```

Department of Computer Science
University of Haifa

החוג למדעי המחשב
אוניברסיטת חיפה

Mount Carmel, 31905 Haifa, Israel Phone: +972-4-8240259 Fax: +972-4-8249331

הר הכרמל, חיפה 31905 טל. 04-8240259 פקס: 04-8249331

Introduction to Computer Science
Homework assignment 2

Submission deadline: 6/11/2002 at midnight

October 27, 2002

1 Quadtaric equation solver

The problem Write a computer program that receives three real numbers: a, b and c, and prints out the solution of the quadratic equation $ax^2 + bx + c = 0$. You should make sure that the input is legal (i.e. make sure a, b and c are real numbers).

How to calculate a square root in C? You should use the *sqrt()* function. *sqrt(x)* will return the square root of x, unless x is a negative number. In order to use *sqrt()* you need to include *math.h* and also link the math library (this is explained further). Here is a short excerpt from the manual:

```
NAME
sqrt - square root function

SYNOPSIS
#include <math.h>

double sqrt(double x);
```

How to link the math library in gcc? Add the switch *-lm* to the compilation commandline. Here is an example:

```
> gcc -Wall -ansi -pedantic -lm myprog.c -o myprog
```

Input/Output Your program will go through automatic checking, so it must take input and give output *exactly* according to the instructions and input/output examples. Assuming we call our executable *quadratic_equation_solver*, and that the commandline prompt is marked with >, please consider the examples in the appendix.

Submission You should submit your homework at the homework submission form at the following URL: http://cs.haifa.ac.il/courses/csintro/upload_form.html. Be sure to select properly the assignment number (you don't want to submit this as any assignment other than 2, right?!), and to properly select the source-code files from your disk. Please name your source-code file *quadratic_equation_solver.c*. Please notice that all students not submitting their exercise before the deadline will receive 0 as their grade.

What to check before submision?

1. Your code should compile with gcc with all the flags we use in class
2. You binary should run and not crash.
3. Your input and output should be *exactly* like the examples on the same input as in the examples.
4. Don't add or reduce anything beyond to what you see in the examples.
5. Your program should also work properly on other values of a, b and c than seen in the examples.
6. If the input is invalid, return *EXIT_FAILURE*, and return *EXIT_SUCCESS* on any other possibility. Invalid input is everything that does not conform to the problem definition. Input that does conform to the problem definition is valid.
7. Submit the source code and not the executable

A Input/Output Examples

In the following example, the values are $a = 0$, $b = 0$ and $c = 0$.

```
> ./quadratic_equation_solver
Please input the coefficients a, b and c for the quadratic equation:
0 0 0
Infinite number of solutions: 0
>
```

In the following example, the values are $a = 1$, $b = 8$ and $c = 15$.

```
> ./quadratic_equation_solver
Please input the coefficients a, b and c for the quadratic equation:
1 8 15
The equation 1.000000X*X+8.000000X+15.000000=0 has two solutions:
-3.000000
-5.000000
>
```

In the following example, the values are $a = 1$, $b = 2$ and $c = 1$.

```
> ./quadratic_equation_solver
Please input the coefficients a, b and c for the quadratic equation:
1 2 1
The equation 1.000000X*X+2.000000X+1.000000=0 has one solution:
-1.000000
>
```

In the following example, the values are $a = 2.55$, $b = 2.55$ and $c = 2.55$.

```
> ./quadratic_equation_solver
Please input the coefficients a, b and c for the quadratic equation:
2.55 2.55 2.55
The equation 2.550000X*X+2.550000X+2.550000=0 has no solution (for real X)
>
```

In the following example, the values are $a = 0$, $b = 0$ and $c = 1$.

```
> ./quadratic_equation_solver
Please input the coefficients a, b and c for the quadratic equation:
0 0 1
No solution
>
```

In the following example, the values are $a = 5 + 4i$, $b = 4.14 - 7i$ and $c = 43$.

```
> ./quadratic_equation_solver
Please input the coefficients a, b and c for the quadratic equation:
5+4i 3.14-7i 43
Problems reading coefficients. Input may be invalid
>
```

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

int main() {
        float a,b,c, delta, solution;

        printf(”Please input the coefficients a, b and c for the quadratic equation:\n”);
        if (3!=scanf(”%f%f%f”,&a,&b,&c)) {
                printf(”Problems reading coefficients. Input may be invalid\n”);
                exit(EXIT_FAILURE);
        }

        if (0==a) {
                if (0==b) {
                        if (0==c) {
                                printf(”Infinite number of solutions\n”);
                        } else {
                                printf(”No solution\n”);
                        }
                } else {
                        solution = -c/b;
                        printf(”The solution is %f\n”,solution);
                }
        } else {
                delta=b*b-4*a*c;
                printf(”The equation %fX*X+%fX+%f=0 has ”,a,b,c);
                if (0>delta) {
                        printf(”no solution (for real X)\n”);
                } else {
                        if (0==delta) {
                                printf(”one solution:\n”);
                                solution = -b/(2*a);
                                printf(”%f\n”,solution);
                        } else {
                                printf(”two solutions:\n”);
                                solution = (-b+sqrt(delta))/(2*a);
                                printf(”%f\n”,solution);
                                solution = (-b-sqrt(delta))/(2*a);
                                printf(”%f\n”,solution);
                        }
                }
        }
        return(EXIT_SUCCESS);
}
```

Department of Computer Science
University of Haifa

החוג למדעי המחשב
אוניברסיטת חיפה

Mount Carmel, 31905 Haifa, Israel Phone: +972-4-8240259 Fax: +972-4-8249331 הר הכרמל, חיפה 31905 טל. 04-8240259 פקס: 04-8249331

Introduction to Computer Science
Homework assignment 4

Submission deadline: Thursday, 21/11/2002 at 7:30am

November 9, 2002

1 C Comments Remover

The problem Write a computer program that reads a valid ANSI-C program from the standard input and prints it to the standard output without the comments. A valid ANSI-C program is a program that compiles with no warnings and without any errors using the *-Wall -ansi -pedantic* flags with the gcc compiler. You may assume your program receives a valid ANSI-C program.

2 Things to pay attention to

You should call your program's source file *remove_c_comments.c* and you should upload only this file as your homework submission. Please DON'T upload any other file. Make sure you upload the file with the correct name.

You may want to read about the common mistakes done in previous homework assignments. This information is available on the course website in the homework section.

Make sure your code compiles with all the flags we saw in class and produces NO errors and NO warnings. Any source file which produces any warning or error during compilation will be graded 0.

Make sure your code works well, try it on the given examples, and also on other examples, and submit only after you are sure it works as you expected it to.

Please document your code with short comments explaining not what you are doing, but why you are doing it.

As usual, late submissions and no submissions will be graded 0. Don't wait too much before you start this exercise, make sure you have enough time to get over all the technical problems, make sure you can submit something. Remember - you can submit your assignment as many times as you like. The LAST submission before the deadline counts as your official submission.

3 Input/Output Examples

Are available on the course website in the homework section.

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

int main() {
        int c;
        int previous = EOF;
        while ( EOF!=(c=getchar()) ) {
                if (c==’”’ && previous!=’\’’) { /* ignore it within strings */
                        do
                                putchar(c);
                        while ( (c = getchar()) != ’”’ && c != EOF );
                } else if (c==’/’) {
                        if ( (c=getchar()) == ’*’ && c != EOF ) {
                                for ( ; ; ) {
                                        while ( (c = getchar()) != ’*’ && c != EOF )
                                        ;    /* eat up text of comment */

                                        if ( c == ’*’ ) {
                                                while ( (c = getchar()) == ’*’ )
                                                ;
                                                if ( c == ’/’ )
                                                        break;    /* found the end */
                                        }

                                        if ( c == EOF ) {
                                                printf(”EOF in comment\n”);
                                                printf(” /* Don’t remove this! */ ”);
                                                break;
                                        }
                                }
                        } else {
                                putchar(’/’);
                        }
                }
                if (c!=EOF && c!=’/’) {
                        putchar(c); /* this is a comment * / inline comment / * *******/
                }
                previous = c;
        }
        return 0;
}
```

Department of Computer Science
University of Haifa

החוג למדעי המחשב
אוניברסיטת חיפה

Mount Carmel, 31905 Haifa, Israel Phone: +972-4-8240259 Fax: +972-4-8249331 הר הכרמל, חיפה 31905 טל. 04-8240259 פקס: 04-8249331

Introduction to Computer Science
Homework assignment 5

Submission deadline: Thursday, 28/11/2002 at 7:30am

November 18, 2002

1 Grades Statistics

The problem Read an unknown number of grades from the standard input (but no more than N grades, where N is some fixed number) and print out statistical information about the grades. The statistical information is, the average, the lowest grade, the highest grade, and a histogram.

The average of K grades is the result of the following calculation:

$$average(grade_0, grade_2, ..., grade_{K-1}) = \frac{1}{K}\sum_{i=0}^{K-1} grade_i$$

The lowest grade of K grades is the result of the following calculation:

$$lowest(grade_0, grade_2, ..., grade_{K-1}) = min(grade_0, grade_2, ..., grade_{K-1})$$

The highest grade of K grades is the result of the following calculation:

$$highest(grade_0, grade_2, ..., grade_{K-1}) = max(grade_0, grade_2, ..., grade_{K-1})$$

The histogram is a bar graph where there is one bar for every 10 numbers starting from 0 and up to 100. The partition of the range of the integer numbers from 0 to 100 should be like this: 0-9, 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99, 100-100. Notice that the last bar contains a range of 1 number while all the rest contain a range of 10 numbers.

A histogram bar contains '*' markers or no markers. If there are no grades in some range than there should be no markers for that range's bar. If there are 9 grades in some range then there should be 9 markers in that range's bar. See published I/O examples for the exact expected formatting.

2 Things to pay attention to

N should be a constant set to 100. Make sure you define N as a constant one time and in one place in the program, as seen in the recitations. Do not use the value 100 explicitly at all when 100 refers to N. You may want to consider defining other constants too.

You should call your program's source file *make_statistics.c* and you should upload only this file as your homework submission. Please DON'T upload any other file. Make sure you upload the file with the correct name.

Your program should read grades until it either reads N grades or encounters an invalid grade. An invalid grade is anything that cannot be read by scanf() as an int and any number not in $[0 - 100]$.

In case the first thing your program reads is an invalid grade, it should print *Invalid input* and halt returning *EXIT_FAILURE*.

In case your program receives N or less grades, it should print out the average, then the lowest grade, then the highest grade and finally the histogram and return *EXIT_SUCCESS*. The average should be printed with exactly two significant digits after the decimal point.

Grades are expected to be integral number in the range $[0 - 100]$. The average should be calculated with double precision. You may assume integral grades.

3 General things to notice

You may want to read about the common mistakes done in previous homework assignments. This information is available on the course website in the homework section.

Make sure your code compiles with all the flags we saw in class and produces NO errors and NO warnings. Any source file which produces any warning or error during compilation will be graded 0.

Make sure your code works well, try it on the given examples, and also on other examples, and submit only after you are sure it works as you expected it to.

Please document your code with short comments explaining not what you are doing, but why you are doing it.

As usual, late submissions and no submissions will be graded 0. Don't wait too much before you start this exercise, make sure you have enough time to get over all the technical problems, make sure you can submit something. Remember - you can submit your assignment as many times as you like. The LAST submission counts as your official submission. Make sure your last submission is before expiration of the submission deadline.

4 Input/Output Examples

Are available on the course website in the homework section.

Use *diff* on study.haifa.ac.il to make sure your output looks EXACTLY like the published examples, and also that your output looks like the output the available binary produces.

```c
/* make_statistics.c
 *
 * Solution to homework assignment #5
 *
 */

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

#define N 100
#define LOWEST_POSSIBLE_GRADE 0
#define HIGHEST_POSSIBLE_GRADE 100
#define BUCKETS 11
#define BUCKET_DIFF 10

#define INVALID_INPUT_MSG "Invalid input"

int main() {
        int min=HIGHEST_POSSIBLE_GRADE;
        int max=LOWEST_POSSIBLE_GRADE;
        double sum=0,average;
        int grade[N];           /* contains the read grades */
        int bucket[BUCKETS]={0};/* counter for each histogram range */
        int i,j;
        int num_grades=0;       /* counts the number of grades read */

        for (i=0; i<N; ++i) {

                if ( (1!=scanf("%d", &grade[i])) ||
                        grade[i]>HIGHEST_POSSIBLE_GRADE ||
                        grade[i]<LOWEST_POSSIBLE_GRADE ) {
                                break; /* no more reading input */
                }

                if (grade[i]<min)
                        min=grade[i];

                if (grade[i]>max)
                        max=grade[i];

                sum+=grade[i];
                bucket[grade[i]/BUCKET_DIFF]++; /* update counter in relevant bucket */
                ++num_grades;
        }

        if (num_grades<1) { /* must have read only invalid input */
                printf("%s\n",INVALID_INPUT_MSG);
                exit(EXIT_FAILURE);
        }

        average=sum/num_grades;

        /* printing the numeric results */
        printf("%.2f\n%d\n%d\n",average,min,max);

        /* printing the histogram */
        for (i=LOWEST_POSSIBLE_GRADE;i<HIGHEST_POSSIBLE_GRADE;i+=BUCKET_DIFF) {
                printf("%3d-%3d:",i,i+BUCKET_DIFF-1);
                for(j=0;j<bucket[i/BUCKET_DIFF];++j)
                        putchar('*');
                putchar('\n');
        }

        /* attending the special case of 100-100 */
        printf("%3d-%3d:",HIGHEST_POSSIBLE_GRADE,HIGHEST_POSSIBLE_GRADE);
        for(j=0;j<bucket[BUCKETS-1];++j)
                putchar('*');
        putchar('\n');
```

```
        return EXIT_SUCCESS;
}
```

Department of Computer Science
University of Haifa

החוג למדעי המחשב
אוניברסיטת חיפה

Mount Carmel, 31905 Haifa, Israel Phone: +972-4-8240259 Fax: +972-4-8249331 הר הכרמל, חיפה 31905 טל. 04-8240259 פקס: 04-8249331

Introduction to Computer Science
Homework assignment 6

Submission deadline: Thursday, 5/12/2002 at 7:30am

November 25, 2002

1 The Goldbach conjecture

The problem A famous conjecture, called the Goldbach conjecture, says that every even integer n greater than 2 has the property that it is the sum of two prime numbers. Computers have been used extensively to test this conjecture. No counterexample has ever been found. Write a program that will prove that the conjecture is true for all the even integers between the two read even integers *start* and *finish*. Of course, due to the definition of the conjecture, the *start* cannot be less than 4.

You get a program called *check_goldbach.c*, which solves this exercise, assuming the implementation (*goldbach.c*) of the interface defined in *goldbach.h* is available.

In other words, you get the main file *check_goldbach.c*, you get the interface file *goldbach.h*, and you also get a *Makefile* (to help you with the compilation); and you need to implement the interface using a file called *goldbach.h*.

As usual, input and output files are available on the course website.

2 Primality testing

Definition: A prime number is an integer number greater than 1, whose only two integral factors are 1 and itself.

The first few prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, 23, and 29.

Think how you implement the primality test *is_prime*. Think about the definition of a prime number, and how you can use it to make sure a number is a prime using the definition. Then, try to look for a better solution, a solution which is more efficient.

3 Things to pay attention to

You should not change the filenames of any of the files given to you. You should implement the interface in a file called *goldbach.c*, and you should upload only this file as your homework submission. Please DON'T upload any other file. Make sure you upload the file with the correct name. The other files given to you are for your use only, and I don't want you to upload any of them to me.

Notice the order of printing, and which number is added to which. To make this easier, use this implementation

```
void print_sum_of_two_primes(int n, int i) {
printf("%d=%d+%d\n",n,i,n-i);
}
```

so now, you have some of the homework solved for you...

4 General things to notice

You may want to read about the common mistakes done in previous homework assignments. This information is available on the course website in the homework section.

Make sure your code compiles with all the flags we saw in class and produces NO errors and NO warnings. Any source file that produces some error during compilation will be graded 0, and source file producing warnings will be penalized according to the number of warnings seen.

Make sure your code works well, try it on the given examples, and also on other examples, and submit only after you are sure it works as you expected it to.

Please document your code with short comments explaining not what you are doing, but why you are doing it.

As usual, late submissions and no submissions will be graded 0. Don't wait too much before you start this exercise, make sure you have enough time to get over all the technical problems, make sure you can submit something. Remember - you can submit your assignment as many times as you like. The LAST submission counts as your official submission. Make sure your last submission is before expiration of the submission deadline.

5 Input/Output Examples

Are available on the course website in the homework section.

Use *diff* on study.haifa.ac.il to make sure your output looks EXACTLY like the published examples, and also that your output looks like the output the available binary produces.

```
#ifndef __GOLDBACH_H__
#define __GOLDBACH_H__

#define FALSE 0
#define TRUE 1
#define INPUT_STRING "Please input two even integers:"
#define COUNTER_EXAMPLE_STRING "is a counter example to Goldbach's conjecture."
#define INVALID_INPUT_STRING "Invalid input: START and FINISH should be positive even integers and START should be less than FINISH."

/*
 * return TRUE if the argument n is a prime number and FALSE otherwise
 */
int is_prime(int n);

/*
 * return TRUE if the argument n is a sum of two prime numbers and FALSE otherwise
 * (remember that if n is the sum of two primes i and n-i, than returning i is
 * like returning a true value (as i cannot be 0)).
 */
int is_sum_of_two_primes(int n);

/*
 * This function is explained in the exercise
 */
void print_sum_of_two_primes(int n, int i);

/*
 * print the argument n and the COUNTER_EXAMPLE_STRING string and exit with EXIT_SUCCESS
 */
void found_counter_example(int n);

void check_conjecture(int from, int to);

#endif
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "goldbach.h"

int is_prime(int n) {
        int i;
        double sqn;
        for (i=2,sqn=sqrt(n); i<=sqn; ++i) {
                if (n%i==0)
                        return FALSE;
        }
        return TRUE;
}

int is_sum_of_two_primes(int n) {
        int i;
        for(i=2; i<n; ++i) {
                if (is_prime(i) && is_prime(n-i))
                        return i; /* any i!=0 is true and i is not 0 */
        }
        return FALSE;
}

void print_sum_of_two_primes(int n, int i) {
        printf("%d=%d+%d\n",n,i,n-i);
}

void found_counter_example(int n) {
        printf("%d %s\n",n,COUNTER_EXAMPLE_STRING);
        exit(EXIT_SUCCESS);
}

void check_conjecture(int from, int to) {
        int n;
        for (n=from; n<to; n+=2) {
                int i;
                if ((i=is_sum_of_two_primes(n))) {
                        print_sum_of_two_primes(n,i);
                } else {
                        found_counter_example(n);
                }
        }
}
```

Department of Computer Science
University of Haifa

החוג למדעי המחשב
אוניברסיטת חיפה

Mount Carmel, 31905 Haifa, Israel Phone: +972-4-8240259 Fax: +972-4-8249331 הר הכרמל, חיפה 31905 טל. 04-8240259 פקס: 04-8249331

Introduction to Computer Science
Homework assignment 7

Submission deadline: Thursday, 12/12/2002 at 7:30am

December 1, 2002

1 The Conway game of life

The Game of Life was invented by John Horton Conway. The game is played on a field of cells, each of which has eight neighbors (adjacent cells). A cell is either occupied (by an organism) or not. The rules for deriving a generation from the previous one are these:

The Rules

- *Death*: If an occupied cell has less than two neighbours (death due to loneliness) or more than three neighbours (death due to overpopulation).
- *Survival*: If an occupied cell has two or three neighbors, the organism survives to the next generation.
- *Birth*: If an unoccupied cell has three occupied neighbors, it becomes occupied.

The game is played on a grid of LIFE_GRID rows and LIFE_GRID columns. The game reads a grid with some organisms on it, this is the 0 generation. Then the rules are being applied iteratively on the current generation's grid and a new generation forms (it might grow or shrink).

As usual, input and output files are available on the course website.

2 The problem

Write a program that reads from the standard input a grid of size LIFE_GRIDxLIFE_GRID characters (LIFE_GRID will be set to 10, meaning, the grids are of size 10x10 cells), where each grid cell is either occupied (the character signifying an occupied cell is a star: '*') or empty (the character signifying an empty cell is a dot: '.'), and employs the game of life on it. The game should run for N generations (N should be set to 20), so that including the 0 generation, there are N+1 generations in total. In every generation your program should print the status of the current grid, and calculate the next generation's grid.

You should implement the *Conway Game of Life* using the *Top Down* method. You will be given the following files:

- The main program file, *conway_game_of_life.c*.

- A header file, *conway.h*, which defines constants and function prototypes of functions used in the main program.
- A makefile, *Makefile*, which defines the dependencies among the files, compilation and linking order.

You should implement the interface described in the header file. The implementation of the functions described in the header file should be in a file called *conway.c*.

3 Assumtions

Input should be LIFE_GRID lines (a line terminates with the newline ('
n') character), each line containing exactly LIFE_GRID characters (not counting the newline). The only two allowed characters in a line are the OCCUPIED character ('*') and the EMPTY character ('.').

The output should be printed to the standard output. Each iteratoin should produce a line stating the number of the generation (starting from the 0 generation), follwed by the grid of that generation. See the published I/O examples for more information.

You might want to define a grid of size (LIFE_GRID+2)x(LIFE_GRID+2), initialize it with an initialization character (e.g. INITIALIZED (the '+' character)), to avoid the trouble in handling the special cases of cells which are on the borders of the original grid. The program should run for N (set to 20) iterations (21 iterations counting the 0 generation too).

The header file *conway.h* contains three functions that you should implement. You should implement these functions in a file called *conway.c*. You might need to implement more functions. Write these too in *conway.c*.

4 General things to notice

You may want to read about the common mistakes done in previous homework assignments. This information is available on the course website in the homework section.

Make sure your code compiles with all the flags we saw in class and produces NO errors and NO warnings. Any source file that produces some error during compilation will be graded 0, and source file producing warnings will be penalized according to the number of warnings seen.

Make sure your code works well, try it on the given examples, and also on other examples, and submit only after you are sure it works as you expected it to.

Please document your code with short comments explaining not what you are doing, but why you are doing it.

As usual, late submissions and no submissions will be graded 0. Don't wait too much before you start this exercise, make sure you have enough time to get over all the technical problems, make sure you can submit something. Remember - you can submit your assignment as many times as you like. The LAST submission counts as your official submission. Make sure your last submission is before expiration of the submission deadline.

5 Input/Output Examples

Are available on the course website in the homework section.

Use *diff* on study.haifa.ac.il to make sure your output looks EXACTLY like the published examples, and also that your output looks like the output the available binary produces.

```
#ifndef __CONWAY_H__
#define __CONWAY_H__

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

#define INVALID_INPUT "Invalid input."
#define EMPTY '.'       /* empty cell */
#define OCCUPIED '*'    /* occupied cell */
#define INITIALIZED '+' /* initialized cell */
#define LIFE_GRID 10
#define REAL_GRID_SIZE ((LIFE_GRID)+2)
#define N 20    /* number of generations the program should simulate. */

void load_init(char[REAL_GRID_SIZE][REAL_GRID_SIZE]);
void print_grid(char[REAL_GRID_SIZE][REAL_GRID_SIZE]);
void step_generation(char[REAL_GRID_SIZE][REAL_GRID_SIZE]);

#endif
```

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

int neighbors(char A[REAL_GRID_SIZE][REAL_GRID_SIZE],int row,int col) {
        int sum=0,i,j;
        for(i=row-1;i<=row+1;++i) {
                for(j=col-1;j<=col+1;++j) {
                        if ((i!=row || j!=col) && OCCUPIED==A[i][j]) {
                                ++sum;
                        }
                }
        }
        return sum;
}

void init_array(char A[REAL_GRID_SIZE][REAL_GRID_SIZE]) {
        int i,j;
        for (i=0;i<REAL_GRID_SIZE;++i)
                for (j=0;j<REAL_GRID_SIZE;++j)
                        A[i][j]=INITIALIZED;
}

void copy_array(char A[REAL_GRID_SIZE][REAL_GRID_SIZE],
                char B[REAL_GRID_SIZE][REAL_GRID_SIZE]) {
        int i,j;
        for (i=1;i<=LIFE_GRID;++i)
                for (j=1;j<=LIFE_GRID;++j)
                        B[i][j]=A[i][j];
}
void sanity_check_array(char A[REAL_GRID_SIZE][REAL_GRID_SIZE]) {
        int i,j;
        for(i=1;i<=LIFE_GRID;++i) {
                for(j=1;j<=LIFE_GRID;++j) {
                        if (INITIALIZED==A[i][j]) {
                                printf("Sanity check detected an unexpected character "
                                        "in row %d and column %d: %c\n", i,j,A[i][j]);
                                exit(EXIT_FAILURE);
                        }
                }
        }
}

void load_init(char A[REAL_GRID_SIZE][REAL_GRID_SIZE]) {
        int row=1,col=1;
        int c;
        init_array(A);
        while( EOF!=(c=getchar())) {
                switch(c) {
                        case ' ':
                        case '\t':
                                break;
                        case '\n':
                                ++row;
                                col=1;
                                break;
                        case EMPTY:
                        case OCCUPIED:
                                A[row][col++]=c;
                                break;
                        default:
                                printf("Invalid input in row %d and column %d: %c\n",
                                                row,col,c);
                                exit(EXIT_FAILURE);
                }
        }
        sanity_check_array(A);
}


void step_generation(char A[REAL_GRID_SIZE][REAL_GRID_SIZE]) {
        int i,j,num_neighbours;
        char Z[REAL_GRID_SIZE][REAL_GRID_SIZE];

        for (i=1;i<=LIFE_GRID;++i) {
                for (j=1;j<=LIFE_GRID;++j) {
```

```
                        num_neighbours=neighbors(A,i,j);
                        if (OCCUPIED==A[i][j]) {
                                /* decide if stays,
                                 * dies from loneliness
                                 * or dies from overpopulation
                                 */
                                Z[i][j]=((2==num_neighbours)||(3==num_neighbours))?OCCUPIED:EMPTY;
                        } else {
                                Z[i][j]=3==num_neighbours?OCCUPIED:EMPTY;
                        }
                }
        }
        copy_array(Z,A);/* Copy the contents of array Z to array A. */
}

void print_grid(char A[REAL_GRID_SIZE][REAL_GRID_SIZE]) {
        int i,j;

        /* Print a row number the actual row */
        for (i=1;i<=LIFE_GRID;++i) {
                for (j=1;j<=LIFE_GRID;++j) {
                        printf("%c ",A[i][j]);
                }
                putchar('\n');
        }
}
```

```
/*
 *
 *      John H. Conway (Scientific American, October 1970, p. 120) invented a game called
 *      Life to model the process of birth, survival, and death.  The idea is that
 *      organisms require others in order to survive and procreate but that overcrowding
 *      results in death.  This program simulates the game Life.
 *      It asks the user for the initial distribution of organisms of the first generation
 *      of the game. After this the game continues for the number of generations defined
 *      by N.  The games follows these rules:
 *
 *      A) Birth Rule: An organism is born into any empty cell that has exactly three
 *              living neighbors.
 *      B) Survival Rule: An organism with either two or three neighbors survives from
 *              one generation to the next.
 *      C) Death Rule: An organism with four or more neighbors dies from overcrowding.
 *              An organism with fewer than two neightbors dies from loneliness.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include ”conway.h”

int main(void) {
        char life[REAL_GRID_SIZE][REAL_GRID_SIZE];      /* Array used for the simulation. */
        int i;

        load_init(life);

        /* Loop to cycle through the different generations. */
        for (i=0;i<=N;i++) {
                printf(”\nGeneration: %i\n”,i); /* print generation number */
                print_grid(life); /* print the current generation */
                step_generation(life);  /* generate the next generation. */
        }
        return EXIT_SUCCESS;
}
```

Department of Computer Science
University of Haifa

החוג למדעי המחשב
אוניברסיטת חיפה

Mount Carmel, 31905 Haifa, Israel Phone: +972-4-8240259 Fax: +972-4-8249331 הר הכרמל, חיפה 31905 טל. 04-8240259 פקס: 04-8249331

Introduction to Computer Science
Homework assignment 9

Submission deadline: Thursday, 26/12/2002 at 7:30am

December 15, 2002

1 Approximated String Matching

In a world where many make mistakes, it is nice to have error correcting mechanisms, especially automatic ones. In this homework we will concentrate on the approximated matching of strings. A known example is a spell cheking and correcting program, which suggests a word upon detecting some erroneous string of characters. Another known example is the mechanism in the Google search engine which suggest a correct term when some erroneous string is input as the search term.

In this home assignments you will implement a program that suggest a word (taken from a dictionary of words) with the smallest edit distance to a given erroneous word.

There are many details in this assignment which you should think about, such as:

- What is a dictionary of words and how can you input it to your program?
- How does the program receive the erroneous word as input?
- How to determine the proximity between words?
- How to find the closest word from a dictionary to the given erroneous word?

Think about these questions when you read the rest of this document. Pay attention to the details of the different definitions. The answers to the questions mentioned above are given in the remainder of this document. Read this text several times, and make sure you understand what you are asked to do. Do no more but no less than what you're required to do.

2 General description of Edit Distance

We shall describe one model (there are several additional possible models) to defining the distance between two strings. Our model deals with converting one string of characters to another string of characters by a series of edit operations on single characters. The allowed operations are:

- INSERT - insertion of a single character to the first string (e.g. CT → CAT).
- DELETE - deletion of a single character from the first string (e.g. CAAT → CAT)

- SUBSTITUTE - substitution of a single character from the first string with a signle character from the other string (e.g. KAT → CAT).

Following are a few examples:

1. Consider the string APPLETS. In order to convert it into the string APPLES we should perform one delete operation.

2. Consider the string CSINTRO. In order to convert it into the string CASINO we need to perform one insert operation (insert the character A) and two delete operation (delete T, delete R).

3. Consider the string BOLD. In order to convert it into the string BALD we need to perform one substitute operation (substitute O with A).

Let A and B be character strings. We define the *edit distance* between A and B as the minimum number of operations (of insert, delete and substitute) needed to be performed on the string A in order to transform it into the string B.

The problem seems difficult, as we need to determine where to delete characters and where to add characters. So, what is the information needed to make the correct decision? For example, what will happen with the last character of the string A? Either a similar character in the same position exists in the string B (and so, there is nothing to do), or we should replace it with another character. Another possibility instead of replacing the character is to delete it and then insert the other character instead. Since we are interested in the minimum number of operations, we will always favour one substitution over the two operations of delete-insert (one delete and one insert).

3 Formal description of Edit Distance

Let A and B be character strings. We define the length of a string A with the number of characters in the string A and we denote this by $|A|$.

We define the distance between two strings A and B in inductively: if A and B are both the empty string, then $edit\_distance(A, B) = 0$
else if B is the empty string, then for any string A $edit\_distance(A, "") = |A|$
else if A is the empty string, then for any string B $edit\_distance("", B) = |B|$
else (so neither A nor B is empty) let the longest prefix of A which does not include the last character of A be pref(A) and let the last character of A be last(A). Similarly, pref(B) and last(B). We define p as:

$$p = \begin{cases} 0 & \text{if } last(A) = last(B), \\ 1 & \text{otherwise.} \end{cases}$$

So, the $edit\_distance(A, B)$ is actually:

$$edit\_distance(A, B) = minimum \left\{ \begin{array}{l} edit\_distance(pref(A), pref(B)) + p, \\ edit\_distance(A, pref(B)) + 1, \\ edit\_distance(pref(A), B) + 1) \end{array} \right\}$$

We say that D_{ij} is the edit distance between the i first characters of A and the j first characters of B. In order to calculate edit_distance(A,B) we need to compute all the values of the matrix D for any $0 \leq i \leq |A|$ and any $0 \leq j \leq |B|$. Therefore, edit_distance(A,B) is the element in the $|A|$th row and the $|B|$th column, i.e. $D_{|A||B|}$.

We need to initialize $D_{0i} = i$ because we want to compare the entire A string to the entire B string, so we will have to dekete the first i characters of A (if we wanted to see whether B is a substring of A we could have demanded a different initial condition: $D_{0i} = 0$). $D_{i0} = i$ because we can't afford deletion of the first i characters with no price.

The algorithm

```
edit_distance(A,B)
    for i=0 to length of A do D[i,0]=i
    for i=0 to length of B do D[0,i]=i
    for i=1 to length of A do
        for j=1 to length of B do
            D[i,j]=minimum(D[i-1,j-1]+p,D[i-1,j]+1,D[i,j-1]+1)
```

4 Things to notice in the implementation

Notice that we need to manage a table (a matrix) of such D_{ij} values (think how many rows and how many colunms such a matrix should have). Such a table can be implemented using two-dimensional arrays. The full size of the table is not required. Two lines of the table are enough: the current line and the previous line.

The table should be dynamically allocated using the lengths of the strings A and B. Since this is the case, you don't need to know the lengths of A and B at all, because you can calculate it once you read the strings from the input. The length of a string can be obtained using the function *strlen()* from *string.h*. Here is a short description of *strlen()*:

```
NAME
strlen - calculate the length of a string

SYNOPSIS
#include <string.h>

size_t strlen(const char *s);

DESCRIPTION
The  strlen()  function  calculates  the  length  of  the string s,
not including the terminating '\0' character.

RETURN VALUE
The strlen() function returns the number of characters in s.
```

When you implement the algorithm make sure you use short and clear functions. You will be required to implement several functions described in a header file *edit_distance.h*.

5 Usage, return values, input/output examples and what to submit

The input to your program comes from the commandline. The input consists of a list of words where the first word is the erroneous word, and the additional words make up the dictionary. The output of your program

should be one word from the dictionary which has the smallest edit distance to the erroneous word. The output should be printed to the standard output.

The usage of the program looks like this:

```
% edit_distance word dict_word1 dict_word2 ... dict_wordN
```

Where *edit_distance* is the name of the executable file, *word* is the erroneous word and *dict_word1* ... *dict_wordN* are N words of the dictionary. Notice that there is no limit on N, so you should expect any non negative integral number of dictionary words. Notice that N is not a constant either!

Here is an example of a possible run:

```
% ./edit_distance applet lake banana rice mouse bottle apple ape
```

And here is the expected output:

```
apple
```

Here is an example of a possible run:

```
% ./edit_distance applet
```

And here is the expected output:

```
Need at least two words
```

You need to implement your solution in a file called *edit_distance.c* which includes *edit_distance.h* (to make sure you follow the prototypes). Submit only *edit_distance.c*.

6 General things to notice

You may want to read about the common mistakes done in previous homework assignments. This information is available on the course website in the homework section. You might find a lot of useful ideas and examples in the recitations preceding this homework assignment.

Make sure your code compiles with all the flags we saw in class and produces NO errors and NO warnings. Any source file that produces some error during compilation will be graded 0, and source file producing warnings will be penalized according to the number of warnings seen. Make sure your code works well, try it on the given examples, and also on other examples, and submit only after you are sure it works as you expected it to.

Please document your code with short comments explaining not what you are doing, but why you are doing it.

As usual, late submissions and no submissions will be graded 0. Don't wait too much before you start this exercise, make sure you have enough time to get over all the technical problems, make sure you can submit something. Remember - you can submit your assignment as many times as you like. The LAST submission counts as your official submission. Make sure your last submission is before expiration of the submission deadline.

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

int minimum( int a, int b, int c ) {
        int min;
        min=(a<b)?a:b;
        min=(min<c)?min:c;
        return min;
}

void memory_allocation_error(void) {
        printf("Memory allocation error. Aborting!\n");
        exit(1);
}

int** alloc_table(int **table, int rows, int cols) {
        int i;
        if (!(table = (int**)malloc(sizeof(int*)*rows)))
                memory_allocation_error();

        for (i=0; i<rows; ++i) {
                if(!(table[i]=(int*)malloc(sizeof(int)*cols)))
                        memory_allocation_error();
        }
        return table;
}

void free_table(int **table, int rows) {
        int i;
        for (i=0; i<rows; ++i) {
                free(table[i]);
        }
        free(table);
}

int** initialize( int** D, int length1, int length2 ) {
        int i,l1,l2;
        l1=length1+1;
        l2=length2+1;
        D = alloc_table(D,l1, l2 );
        for (i=0; i<l1; ++i) {
                D[i][0]=i;
        }
        for (i=0; i<l2; ++i) {
                D[0][i]=i;
        }
        return D;
}

int edit_distance(char *word1, char *word2) {
        int i, j, p, ret;
        int **D;
        int length1, length2;
        length1=strlen(word1);
        length2=strlen(word2);
        D = initialize(D,length1,length2);
        for (i=1; i<=length1; ++i) {
                for (j=1; j<=length2; ++j) {
                        p = (word1[i-1]!=word2[j-1]);
                        D[i][j]=minimum( D[i-1][j-1]+p, D[i-1][j]+1, D[i][j-1]+1 );
                }
        }
        ret = D[--i][--j];
        free_table(D,length1);
        return ret;
}
```

```
void check_input(int num_params) {
        if (num_params<3) {
                printf("Need at least 2 arguments.\n");
                exit(EXIT_FAILURE);
        }
}

char* get_closest(char *word1, char *dict_words[], int dict_size) {
        int i,d,closest;
        int min=INT_MAX;
        for (i=0,closest=0; i<dict_size; ++i) {
                d=edit_distance(word1,dict_words[i]);
                if (min>d) {
                        min=d;
                        closest=i;
                }
        }
        return dict_words[closest];
}

void print_closest(char *word) {
        printf("%s\n",word);
}

int main( int argc, char *argv[] ) {
        check_input(argc);
        print_closest(get_closest(argv[1],2+argv,argc-2));
        return EXIT_SUCCESS;
}
```

Department of Computer Science
University of Haifa

החוג למדעי המחשב
אוניברסיטת חיפה

Mount Carmel, 31905 Haifa, Israel Phone: +972-4-8240259 Fax: +972-4-8249331 הר הכרמל, חיפה 31905 טל. 04-8240259 פקס: 04-8249331

Introduction to Computer Science
Homework assignment 10

Submission deadline: Thursday, 2/1/2003 at 7:30am

December 22, 2002

1 Recursive above-average

Write a function *above_average()* with the following prototype:

```
double above_average(int sum, int n)
```

reads integer numbers from the standard input and then prints in reversed order all the numbers that are greater than the average of the numbers seen in input.

You should submit the function's implementation in a filename called *above_average.c*. This file should includ the header file *above_average.h*

You can test your implementation using the test driver *test_above_average.c*

The files *above_average.h* and *test_above_average.c* are published in the homework section of the course site. You should only submit *above_average.c*, completing it. Notice that you are not allowed to use any other memory and variable other than those already defined in *above_average.c*.

2 General things to notice

You may want to read about the common mistakes done in previous homework assignments. This information is available on the course website in the homework section. You might find a lot of useful ideas and examples in the recitations preceding this homework assignment.

Make sure your code compiles with all the flags we saw in class and produces NO errors and NO warnings. Any source file that produces some error during compilation will be graded 0, and source file producing warnings will be penalized according to the number of warnings seen. Make sure your code works well, try it on the given examples, and also on other examples, and submit only after you are sure it works as you expected it to.

Please document your code with short comments explaining not what you are doing, but why you are doing it.

As usual, late submissions and no submissions will be graded 0. Don't wait too much before you start this exercise, make sure you have enough time to get over all the technical problems, make sure you can submit something. Remember - you can submit your assignment as many times as you like. The LAST submission counts as your official submission. Make sure your last submission is before expiration of the submission deadline.

```
#include <stdio.h>
#include ”above_average.h”
double above_average(int sum, int n) {
        double average;
        int input;

        if (scanf(”%d”, &input) != 1) {
                return sum/(double)n;
        } else {
                average=above_average(sum+input, n+1);
                if (average < input) {
                        printf (”%d ”, input);
                }
                return average;
        }
}
```

תרגיל בית 11

עצה: אל תתחילו לעשות דבר בתרגיל-הבית לפני שקראתם את התרגיל בעיון לפחות פעם אחת מההתחלה ועד הסוף.

חלוקה נכונה של הבעיה לתת-בעיות קטנות יותר, תקל עליכם הן בהגדרת הפונקציות שלכם והן בפתרון מלא ונכון של הבעיה תוך מימוש פתרונכם בשפת C.

מוטיבציה:

יישומים רבים לניתוח אוטומטי של טקסטים עושים שימוש בנתונים סטטיסטיים הנאספים מתוך הטקסטים. דוגמא לנתון סטטיסטי כזה הוא מספר המופעים של כל מילה בטקסט נתון, שכן מסתבר כי ניתן להסיק מסקנות רבות משכיחותה של מילה כזאת או אחרת.

בתרגיל זה אנו נעשה שימוש במבנה הנתונים TRIE כדי לאחסן את המילים אשר נראו בטקסט. כמו-כן נשמור עבור כל מילה במבנה הנתונים את מספר המופעים שלה בטקסט הנקרא.

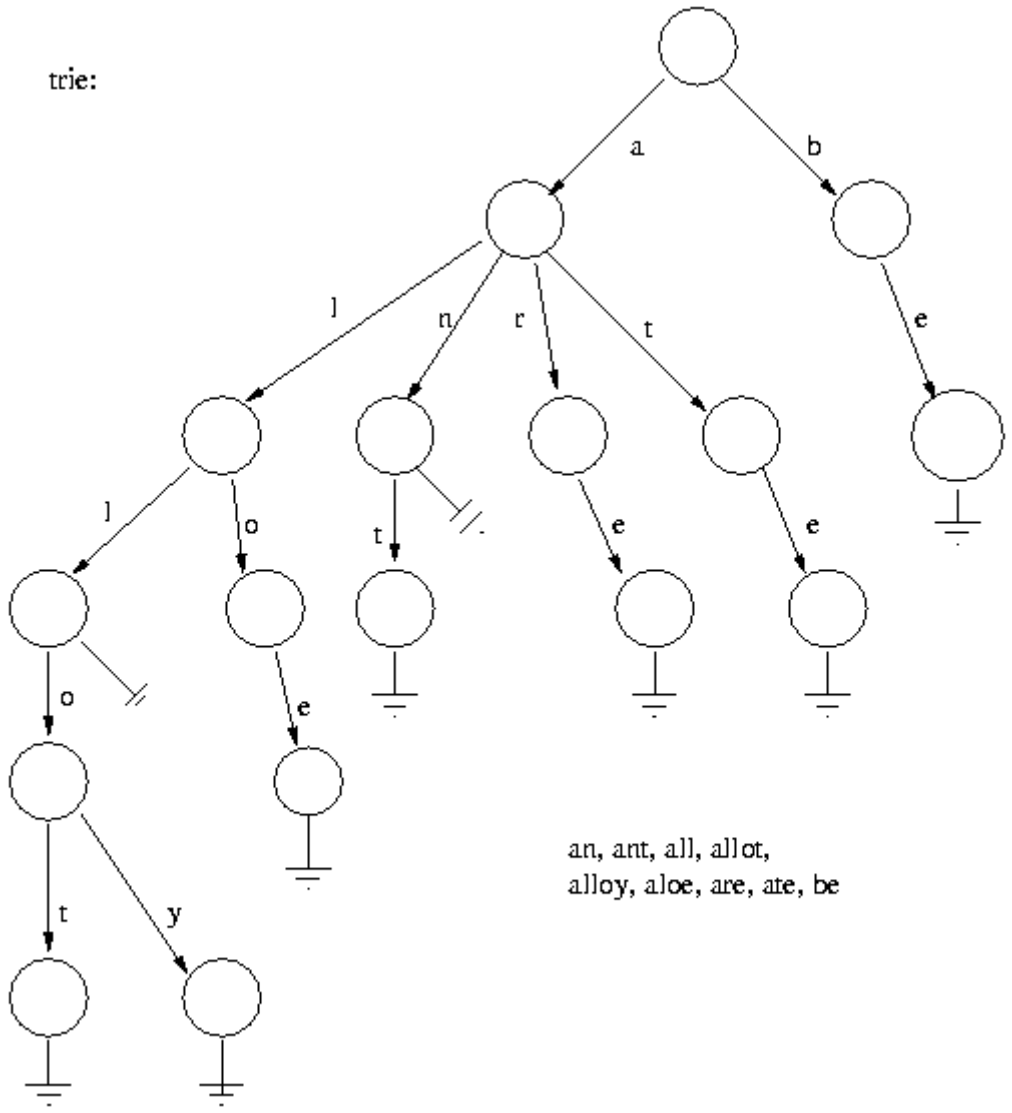
ל-TRIE יתרון באחסון אוספים של מילים, בפרט, ניתן לאחסן ביעילות טובה מאד מילים בעלות רישא (תחילית) משותפת.

עליכם יהיה להגדיר את המבנים של צומת בעץ (ואולי גם מבנים נוספים) וכן עליכם לממש את פעולות יצירת העץ והריסתו, גידול העץ על-ידי הכנסת מילים, תחזוק מוני תדירות של מילים וכיוצא באלה.

כתב: שלמה יונה shlomo@cs.haifa.ac.il.
ניתן להשיג את תרגילי הבית באתר הקורס – http://www.cs.haifa.ac.il/courses/csintro
פתרונות לתרגיל יופיעו באתר הקורס לאחר מועד הגשה האחרון.

ה-TRIE:

נגדיר מבנה נתונים TRIE. ה-TRIE הוא עץ בו מאחסנים מחרוזות. נתבונן בדוגמא הבאה בה אנו מאחסנים את המילים: an, ant, all, allot, alloy, aloe, are, ate, be. ה-TRIE המכיל את המילים הללו נראה כך:



כל צומת בעץ - או שיש לו ילדים, או שאין לו ילדים (צומת חסר ילדים נקרא *עלה*).
TRIE הוא עץ בו כל קשת היוצאת מצומת מסומנת על-ידי אות באלף-בית נתון (אנו נשתמש באלף-בית המכיל את כל האותיות באנגלית a-z). אין שתי קשתות היוצאות מצומת משותף ומסומנות באותה האות. נשים לב כי מילים שלהן רישא משותפת חולקות תת-עץ, שימו לב בדוגמא למשל למילים all ו-allot.

כתב: שלמה יונה shlomo@cs.haifa.ac.il.

המשימה:

עליכם לכתוב תכנית אשר תקרא מילים מתוך ה-standard input ותדפיס את רשימת המילים אשר נראו בקלט עם מספר המופעים של כל מילה ומילה. את הרשימה יש להדפיס, כתלות בפרמטר משורת הפקודה, באחד מן האופנים הבאים:

- רשימה ממוינת לפי סדר לקסיקוגרפי עולה (ברירת המחדל)
- רשימה ממוינת לפי סדר לקסיקוגרפי יורד (לפי פרמטר).

הגדרת הרצת התכנית:

./frequency
תדפיס את רשימת המילים שנראו בקלט יחד עם מספר המופעים לכל מילה ממוינת בסדר לקסיקוגרפי עולה.

./frequency r
תדפיס את רשימת המילים שנראו בקלט יחד עם מספר המופעים לכל מילה ממוינת בסדר לקסיקוגרפי יורד.

יש להדפיס הודעת שגיאה ולצאת עם קוד חזרה EXIT_FAILURE, במקרה ושורת הפקודה אינה כמתואר לעיל.

דגשים במימוש:

אנו נתמוך רק באותיות קטנות לכן עליכם לדאוג להמרה של אותיות גדולות לאותיות קטנות.

עליכם להתעלם מכל סימן בקלט שאינו אחת מעשרים ושש האותיות הקטנות באנגלית.

את הפלט עליכם להדפיס באופן הבא: כל מילה מופיעה בשורה נפרדת. הסימן tab יפריד בין מילה לבין מספר המופעים שלה. למשל, עבור הקלט
abc bac bac abc ddd aaa.
ועבור הרצה ללא פרמטרים בשורת הפקודה, נצפה לפלט הבא:

```
aaa	1
abc	2
bac	2
ddd	1
```

זכרו לשחרר זיכרון שהוקצה על-ידכם באופן דינאמי לאחר שסיימתם את השימוש בזיכרון.

כתב: שלמה יונה shlomo@cs.haifa.ac.il.

הצעת הגדרות לשימושכם (אינכם חייבים להשתמש):

```
#define NUM_LETTERS ((int)26)
```

באלף-בית האנגלי יש עשרים ושש אותיות.

```
typedef enum {FALSE=0, TRUE=1} boolean;
```

נגדיר טיפוס בינארי לאמת ושקר.

```
typedef struct node {
      char letter;
      long unsigned int count;
      struct node* children[NUM_LETTERS];
} node;
```

צומת יכיל את

- האות המופיעה על הקשת הנכנסת אליו
- מונה לספירת מספר מילים
- מערך בן עשרים ושישה מצביעים לצומת, עבור ילדים פוטנציאליים.

אתם רשאים להוסיף שדות נוספים, אם תרצו או להשתמש במבנים אחרים לפי הבנתכם. חישבו למשל, כיצד תדעו האם לצומת יש או אין ילדים. חישבו כיצד תדעו האם צומת מסמן גם סוף של מילה. בכל מקרה, כאשר אתם מגדירים את הצומת שלכם, על שדותיו, הקפידו לתעד כל הנחה וכל החלטה שאתם מקבלים.

זיכרו: כל ערך קבוע אשר אתם עושים בו שימוש, רצוי להגדירו במקום מסודר בתחילת התכנית (למשל על ידי `#define`).

הקפידו על ניהול זכרון נכון (הקצאה ושחרור).

בהצלחה!

כתב: שלמה יונה shlomo@cs.haifa.ac.il.
ניתן להשיג את תרגילי הבית באתר הקורס – http://www.cs.haifa.ac.il/courses/csintro
פתרונות לתרגיל יופיעו באתר הקורס לאחר מועד הגשה האחרון.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define NUM_LETTERS ((int)26)
typedef enum {NO=0, YES=1} boolean;

typedef struct node {
        char letter;
        long unsigned int count;
        boolean is_word;
        boolean has_kids;
        struct node* children[NUM_LETTERS];
} node;

typedef struct trie {
        node* children[NUM_LETTERS];
        node* current;
        boolean empty;
        int max_word_length;
        char* word;
} trie;

void memory_allocation_error(void) {
        printf("Memory allocation error. Aborting!\n");
        exit(1);
}

node* new_node(void) {
        node* n;
        if (!(n=(node*)malloc(sizeof(node))))
                memory_allocation_error();
        return n;
}

node* initialize_node(node* n,char letter) {
        int i;
        n->letter = letter;
        n->count = 0;
        n->is_word = NO;
        n->has_kids = NO;
        for (i=0; i<NUM_LETTERS; ++i) {
                (n->children)[i] = NULL;
        }
        return n;
}

node* create_node(char letter) {
        node* created_node;
        created_node=new_node();
        return initialize_node(created_node,letter);
}

void free_node(node* n) {
        int i;
        if (n==NULL) {
                return;
        }
        if (0==(n->has_kids)) {
                free(n);
        } else {
                for (i=0; i<NUM_LETTERS; ++i) {
                        free_node((n->children)[i]);
                }
        }
        return;
}

trie* new_trie(void) {
        trie* t;
```

```
        if (!(t=(trie*)malloc(sizeof(trie))))
                memory_allocation_error();
        return t;
}

trie* initialize_trie(trie* t) {
        int i;
        for (i=0; i<NUM_LETTERS; ++i) {
                t->children[i]=NULL;
        }
        t->current=NULL;
        t->empty=YES;
        t->max_word_length=0;
        return t;
}

trie* create_trie(void) {
        trie* created_trie;
        created_trie=new_trie();
        return initialize_trie(created_trie);
}

void close_word(trie* root) {
        if (root->current == NULL)
                return;
        root->current->count++;
        root->current->is_word = YES;
        root->current=NULL;
}

boolean is_empty(trie* root) {
        return root->empty;
}

int char2index(char c) {
        return c-'a';
}

int read_character(trie* root, int c) {
        int index;
        int word_length=0;
        if(!isalpha(c)) {
                close_word(root);
                return word_length;
        }
        word_length++;
        c=tolower(c);
        index= char2index(c);
        if (root->current==NULL) { /* new word  - start from root */
                if (root->children[index] == NULL)
                        root->children[index] = create_node(c);
                root->current = root->children[index];
                root->empty=NO;
        } else {
                root->current->has_kids = YES;
                if (root->current->children[index] == NULL)
                        root->current->children[index] = create_node(c);
                root->current=root->current->children[index];
        }
        return word_length;
}

void allocate_word(trie* root) {
        free(root->word);
        if (!(root->word=(char*)malloc(1+sizeof(char)*(root->max_word_length))) )
                memory_allocation_error();
}

trie* read_text() {
```

```
        int c;
        int word_length;
        trie* root;
        root = create_trie();
        while( EOF!=(c=getchar()) ) {
                word_length=read_character(root,c);
                if (word_length>root->max_word_length)
                        root->max_word_length=word_length;
        }
        allocate_word(root);
        return root;
}

void print_words_reverse(trie* root) {
        static int p=0;
        int i;
        node* current;
        root->word[p++]=root->current->letter;
        if (root->current->has_kids) {
                for (i=NUM_LETTERS-1; i>=0; --i) {
                        if (root->current->children[i] == NULL)
                                continue;
                        current = root->current; /* remember */
                        root->current = root->current->children[i];
                        print_words_reverse(root);
                        root->current = current; /* recall */
                }
        } else {
                if (root->current->is_word) {
                        root->word[p]='\0';
                        printf("%s\t%ld\n",root->word,root->current->count);
                }
                --p;
                return;
        }
        if (root->current->is_word) {
                root->word[p]='\0';
                printf("%s\t%ld\n",root->word,root->current->count);
        }
        --p;
}

void print_trie_reverse(trie* root) {
        int i;
        if (root == NULL)
                return;
        if (is_empty(root))
                return;
        for (i=NUM_LETTERS-1; i>=0; --i) {
                if (root->children[i] == NULL)
                        continue;
                root->current = root->children[i];
                print_words_reverse(root);
        }
}

void print_words(trie* root) {
        static int p=0;
        int i;
        node* current;
        root->word[p++]=root->current->letter;
        if (root->current->is_word) {
                root->word[p]='\0';
                printf("%s\t%ld\n",root->word,root->current->count);
        }
        if (root->current->has_kids) {
                for(i=0; i<NUM_LETTERS; ++i) {
                        if (root->current->children[i] == NULL)
                                continue;
```

```
                        current = root->current; /* remember */
                        root->current = root->current->children[i];
                        print_words(root);
                        root->current = current; /* recall */
                }
        } else {
                --p;
                return;
        }
        --p;
}

void print_trie(trie* root) {
        int i;
        if (root == NULL)
                return;
        if (is_empty(root))
                return;
        for (i=0; i<NUM_LETTERS; ++i) {
                if (root->children[i] == NULL)
                        continue;
                root->current = root->children[i];
                print_words(root);
        }
}

void free_trie(trie* t) {
        int i;
        if (t == NULL)
                return;
        for(i=0;i<NUM_LETTERS; ++i) {
                free_node(t->children[i]);
        }
        free(t);
}

void usage(char* program_name,char* message) {
        printf("\n%s\n\n",message);
        printf("USAGE:\n\t%s: %s\n\n",program_name,message);
        exit(1);
}

boolean should_reverse(int argc, char* argv[]) {
        if (argc > 2)
                usage(argv[0],"Wrong number of arguments.");
        if ( (argc == 2) && (argv[1][0]=='r' || argv[1][0]=='R') )
                return YES;
        if (argc == 1)
                return NO;
        usage(argv[0],"Bad command line arguments.");
        return NO; /* will never get here */
}

int main(int argc, char* argv[]) {
        trie* root;
        boolean r=NO;
        r = should_reverse(argc,argv);
        root = read_text();
        if (r)
                print_trie_reverse(root);
        else
                print_trie(root);
        free_trie(root);
        return 0;
}
```