תרגיל מס׳ 6 - C – מבנים והקצאה דינמית (מתרגל אחראי: אייל)

התרגיל שווה 25% מציון התרגול

הוראות הגשה

שאלות בנוגע לתרגיל נא להפנות דרך פורום הקורס שנפתח במיוחד לשם כך:
https://piazza.com/biu.ac.il/fall2017/89110/.

אם לא נענתה תשובה תוך 24 שעות, נא לשלוח אלי (אייל) מייל עם לינק לדיון הרלוונטי ואענה. המייל הוא: eyal.dayan@live.biu.ac.il. בכל מייל יש לציין **שם, שם משתמש, מס' קורס, וקבוצת תרגול.**

- מועד פרסום: 28/12/17
- מועד אחרון להגשה: 11/1/18 23:50
- יש לשלוח את הקבצים באמצעות מערכת ההגשה לפני חלוף התאריך הנקוב לעיל.
- אין הארכות (למעט מילואים), אך ניתן להגיש באיחור עם קנס (עד יומיים).
  יום איחור גורר הורדה אוטומטית של 10 נקודות.
  יומיים איחור גוררים הורדה אוטומטית של 20 נקודות.

- שם ההגשה של התרגיל: ex6
- יש להקפיד מאוד על כל הוראות עיצוב הקלט והפלט, כמפורט בכל סעיף וסעיף. על הפלט להיראות בדיוק כמו בדוגמאות. **אין להוסיף או להשמיט רווחים או תווים אחרים ואין להחליף אותיות גדולות בקטנות או להיפך. אי-הקפדה על פרטים אלה עלולה לגרור ירידה משמעותית ביותר בציון התרגיל עד כדי 0. ראו הוזהרתם!**
  טיפ - אפשר להגיש קובץ ריק למערכת ההגשה ולהעתיק מהמייל החוזר את מחרוזות הפלט. אל תעתיקו מתוך מסמכי Pdf ומצגות.
- להזכירכם, העבודה היא אישית. "עבודה משותפת" דינה כהעתקה.
- אין להדפיס למסך שום דבר מעבר למה שנתבקש בתרגיל.
- יש לוודא שהתרגיל מתקמפל ורץ על השרתים באוניברסיטה (u2) ללא שגיאות/אזהרות.
- אתם יכולים לעבוד עם כל עורך טקסטואלי שאתם מעדיפים. להזכירכם – pico בשרתי linux שבמעבדות; notepad ב-windows ; או בסביבת פיתוח ייעודית.

הקפידו על כתיבה לפי קובץ ה-Coding-Style שבאתר הקורס!

הקדמה

- בתרגיל זה אין הנחות מוקדמות לגבי כמות הזיכרון שיש להקצות, למעט לגבי אורכן של מחרוזות. עם זאת, לא נרצה להקצות זיכרון מבוזבז, ולכן כל המידע חייב להימצא ב-heap באופן דינמי. חשוב לעבוד באופן מסודר, ולכן התרגיל מחולק למודולים. זכרו להימנע מדליפות זיכרון ומעבודה לא חוקית עם מצביעים.

בתרגיל זה נממש "משחק" בהשראת בעיית יוספוס. ספציפית, הגרסה שנממש היא בהשראת הסרטון הזה, לכן מומלץ לצפות בו (מי שלא רוצה לא חייב להתעמק יותר מדי במתמטיקה ובהוכחות, החלק הרלוונטי הוא בעיקר הסיפור).

התרגיל מחולק לקבצים הבאים:
main.c - קובץ מקור שיכיל פונקציית main בלבד.
המודול **game** המורכב מהקבצים **game.h game.c**
המודול **person** המורכב מהקבצים **person.h person.c**
המודול **sorting** המורכב מהקבצים **sorting.h sorting.c**
המודול **utils** המורכב מהקבצים **utils.h utils.c**

- המודול **game** מכיל את הלוגיקה הכללית של המשחק (בעיקר מבחינת העבודה עם מבנה הנתונים רשימה מקושרת).
- המודול **person** מכיל מבנה (struct) המייצג אדם. לכל אדם יש שם פרטי, מספר ת"ז, ויתכן גם שיהיו לו ילדים. בנוסף המודול מכיל פונקציות שרלוונטיות לצורך טיפול במבנה של אדם.
- המודול **sorting** מכיל פונקציות שממיינות מעגל נתון.
- המודול **utils** מיועד לפונקציות עזר שמשתמשים בהן בכמה מודולים שונים, ונועד לעזור לכם לעשות סדר.

הסבר מפורט על כל מודול יופיע בהמשך המסמך.

פקודת הקימפול למי שבודק על השרת:
gcc main.c game.c game.h person.c person.h sorting.c sorting.h utils.c utils.h -std=c99

כל קובץ h בתוכנית שלנו יכיל התניית ifndef-define-endif על מנת למנוע קומפילציה כפולה של המודול במידה ואנו כוללים את הקובץ בתוך כמה קבצים אחרים (יותר מאחד).

כל הקבצים המוצגים במסמך זה מצורפים לתרגיל, לכן אין צורך להעתיק מתוך המסמך (ובאופן כללי לא מומלץ להעתיק קוד מתוך מסמכי pdf).

הוראות הגשה

אין לשנות (ולכן גם אין צורך להגיש) את הקבצים הבאים: person.h sorting.h game.h

את הקובץ utils.h ניתן להרחיב, אך לא למחוק ממנו את מה שכבר מופיע בו.

אין צורך להגיש את הקובץ main.c (ולכן אתם מוזמנים לשנות אותו כרצונכם לצורכי בדיקה).

לסיכום, עליכם להגיש את חמשת הקבצים הבאים: person.c sorting.c game.c utils.c utils.h

המודול Person

הקובץ person.h נראה כך:

```
#ifndef PERSON_H
#define PERSON_H

struct Person {
   char* name;
   int id;
   int numOfKids;
   char** kids;
   struct Person* next;
   void (*Print)(struct Person* this);
   void (*KillNext)(struct Person* this);
   struct Person* (*SelfDestruct)(struct Person* this);
   void (*CreateNext)(struct Person* this, int isNextNext, ...);
};
typedef struct Person Person;

//use only once
Person* CreatePerson();
void InitPersonValues(Person* person);

#endif
```

המבנה person מכיל מצביע לשמו הפרטי (המוקצה דינמית, על מנת לחסוך בזיכרון. אין לשנות את ההגדרה למערך של תווים בגודל ידוע מראש, למרות שיינתן בהמשך חסם על אורך של שם בתוכנית), מספר מזהה ייחודי, מספר ילדיו, מצביע למערך של שמות ילדיו, ומצביע לאדם הבא במעגל. בנוסף מכיל כל אדם מצביעים לפונקציות הבאות:

Print – הפונקציה מדפיסה אדם אל המסך (דוגמה בסוף התרגיל).

KillNext – הפונקציה הורגת את האדם שנמצא אחרי האדם שממנו הופעלה הפונקציה, ומשחררת את הזיכרון.

SelfDestruct – הפונקציה משחררת את הזיכרון של האדם שממנו הופעלה הפונקציה.

CreateNext – הפונקציה מקצה זיכרון עבור אדם חדש וממקמת אותו לאחר האדם שממנו הופעלה הפונקציה. במידה והיה שם כבר אדם קודם, הפונקציה תמקם אותו לאחר האדם שזה עתה נוסף. הפרמטר השני של הפונקציה הוא בוליאני: 1 אם קיים אדם כזה (סיטואציה כזו תתרחש כשמנסים להוסיף מישהו שאיחר למעגל קיים), 0 אם לא קיים אדם כזה (סיטואציה כזו תתרחש במהלך הבנייה של המעגל). לאחר מכן מופיעות שלוש נקודות במקום פרמטר שלישי - אין צורך להיבהל, מדובר בהגדרה פשוטה שמאפשרת לפונקציה לקבל מספר משתנה של פרמטרים, בדומה לפונקציה printf. בסוף התרגיל מופיע נספח עם דוגמה פשוטה לפונקציה כזו.

כל אחת מהפונקציות האלו מקבלת כפרמטר ראשון משתנה מסוג מצביע לperson. כמוסכמה, בכל פעם שנפעיל פונקציה מתוך מבנה ספציפי, נשלח לו את אותו המבנה כפרמטר ראשון (אין להפעיל את הפונקציות האלו באופן אחר). היינו שמחים כמובן אם הפונקציה הייתה יכולה "לדעת לבד" איזה מבנה ביצע את הקריאה, אך זה לא אפשרי בשפת c.
לדוגמה, נניח כי person הוא מצביע למבנה מסוג Person, אז הדפסה שלו תתבצע כך:

```
person->Print(person);
```

בנוסף, מכיל המודול שתי פונקציות נוספות על מנת לאפשר יצירה של האדם הראשון (אין להשתמש בפונקציות האלו מחוץ למודול מעבר לכך, אלא רק בפונקציות מתוך מבנים שכבר קיימים, באופן שבו הוסבר קודם).

הפונקציה CreatePerson מקצה את הזיכרון עבור האדם הראשון ומחזירה את המצביע אליו.
הפונקציה InitPersonValues מאתחלת את כל השדות בתוך המבנה החדש (כולל המצביעים לפונקציות).

- ניתן להסיק מכך, שהפונקציה CreateNext תבצע בסך הכל קריאה לפונקציה CreatePerson (שתקרא בתחילתה לפונקציה InitPersonValues), ותדאג למקם את המבנה החדש שנוצר במקום הנכון.

בקובץ ה-c יש לממש את הפונקציות האלו, ושם גם יהיה מקומן של הפונקציות Print, KillNext, SelfDestruct, CreateNext אלא שאין צורך להחצין אותן משום שניתן להשתמש בהן דרך כל מבנה שמאותחל כראוי.

המודול Game

הקובץ game.h נראה כך:

```
#ifndef GAME_H
#define GAME_H

#include "person.h"

Person* InitTheHungerGame();
void InsertLaters(Person* head);
Person* RemoveCowards(Person* head);
void LetTheHungerGameBegin(Person* head);

#endif
```

הקובץ כולל הכללה של המודול person משום שמוגדרות בו פונקציות שצריכות להכיר את המודול הזה.
הקובץ מגדיר את הפונקציות הבאות:

InitTheHungerGame – הפונקציה מאתחלת את המשחק. לאחר הקריאה יוחזר מצביע אל האדם הראשון מתוך מעגל שהוקצה באופן דינמי. במקרה של כישלון, או שאין אנשים כלל הפונקציה תחזיר NULL.
InsertLaters – הפונקציה מכניסה אנשים נוספים שאיחרו להגיע, ומבקשים להיכנס למעגל ולהתמקם בו לאחר החבר הכי טוב שלהם. שימו לב שאין ערך החזרה משום שראש המעגל לא יכול להשתנות כתוצאה מכך. כמו כן, לא ניתן להצטרף למעגל ריק.
RemoveCowards – הפונקציה מוציאה מהמעגל את הפחדנים שאין בהם את האומץ הדרוש על מנת להשתתף במשחק. שימו לב שהפונקציה מחזירה מצביע חדש משום שייתכן והסרנו את האדם הראשון. במידה וכל האנשים פחדנים הפונקציה תחזיר NULL, כמובן שלא ניתן להסיר אנשים ממעגל ריק.
LetTheHungerGameBegin – הפונקציה מתחילה את המשחק ובסופו מודיעה מי ניצח ונשאר בחיים. כמובן שאין משחק אם המעגל ריק, ובמעגל בגודל 1 המשחק קצר מאוד (ניצחון מיידי).

המודול Sorting

הקובץ sorting.h נראה כך:

```
#ifndef SORTING_H
#define SORTING_H

#include "person.h"

Person* SortCircleByID(Person* head);
Person* SortCircleByName(Person* head);

#endif
```

הקובץ כולל שתי פונקציות המקבלות מעגל וממיינות אותו לפי שמות האנשים או לפי המספרים המזהים שלהם. הפונקציות מחזירות מצביע לראש המעגל החדש. הפונקציות לא מדפיסות דבר.

ניתן להשתמש באלגוריתם המיון שהוצג בכיתה, אך יש להימנע משכפול קוד מיותר.

המודול Utils

הקובץ utils.h נראה כך:

```
#ifndef UTILS_H
#define UTILS_H

#include "person.h"

void PrintCircle(Person* head);
//more functions declarations

#endif
```

הקובץ יכיל פונקציות שבהן משתמשים בכמה מקומות שונים, וכן פונקציות עזר שסתם לא מצאתם עבורן מקום הגיוני באחד מהמודולים הקיימים.

כרגע המודול מכיל את הפונקציה PrintCircle המקבלת מצביע לראש המעגל ומדפיסה אותו (דוגמה בסוף התרגיל).

דוגמה לקובץ main

```
#include "person.h"
#include "utils.h"
#include "sorting.h"
#include "game.h"

int main() {
   Person* head = InitTheHungerGame();
   PrintCircle(head);
   InsertLaters(head);
   head = RemoveCowards(head);
   head = SortCircleByID(head);
   PrintCircle(head);
   head = SortCircleByName(head);
   PrintCircle(head);
   LetTheHungerGameBegin(head);
   return 0;
}
```

בתחילת התוכנית מתבצעת קריאה לאתחול המשחק, ובסופה מתבצעת קריאה להתחלת המשחק. קריאה לפונקציית האתחול למעשה תיצור מספר מעגלים בלתי תלויים, ואם לא תתבצע קריאה להתחלת המשחק עבור כל מעגל, תיווצר דליפת זיכרון (וזה בסדר, משום שכך הוגדר המודול. יש להסביר זאת בתיעוד הפונקציות כמובן). בכל שלב ניתן לקרוא לפונקציות SortCircleByID \ SortCircleByName הממיינות את המעגל לפי שם / ת"ז, או לפונקציה PrintCircle המדפיסה את המעגל. דוגמאות הרצה יופיעו בסוף התרגיל.

שימו לב, הפונקציה InitTheHungerGame מקצה זיכרון, והפונקציה LetTheHungerGameBegin תשחרר את הזיכרון הזה בסוף התוכנית. הפונקציה InsertLaters היא מעין תוספת למשימת ההקצאה, היא מבצעת חיפוש במעגל הקיים (ייתכן שתרצו לכתוב פונקציית עזר לשם כך, מקומה יהיה במודול utils) ומבצעת הקצאה נוספת. הפונקציה RemoveCowards היא מעין תוספת למשימת השחרור, היא מבצעת חיפוש (כנ"ל, פונקציית עזר – אם כי היא מעט שונה) ומשחררת זיכרון. פונקציות המיון וההדפסה לא מקצות זיכרון שנשאר לאחר פעולתן (כמובן שהן יכולות להקצות זיכרון ולשחרר אותו אם יש צורך בכך).

בנוסף, שימו לב שחלק מהפונקציות מחזירות מצביע חדש משום שראש המעגל עלול להיות שונה לאחר פעולתן. ככלל, מומלץ להגדיר את ההתנהגות של פונקציות ההקצאה בדומה לפונקציה malloc – מחזירות מצביע תקין במקרה של הצלחה, ומחזירות NULL במידה והן נכשלות (כלומר לא קיימת מבחינתן סיטואציה של "הצלחה חלקית" שבה הן מבצעות רק חלק מהמשימה משום שקרתה תקלה באמצע, למשל כישלון בהקצאת הזיכרון עבור האדם החמישי – במקרה כזה הפונקציה InitTheHungerGame תשחרר את כל הזיכרון שהוקצה ותחזיר NULL). את ההתנהגות של פונקציות השחרור הגדירו באופן שונה מהאופן שבה מוגדרת הפונקציה free – נרצה שהתוכנית שלנו תהיה נחמדה יותר, ולכן בעת העברה של מצביע בלתי חוקי הן פשוט לא יבצעו דבר.

קבצי C

קבצי ה-c יכילו מימושים להצהרות המופיעות בקבצי ה-h. ניתן להשתמש בספריות malloc, string, stdio ורק בפונקציות שנלמדו בתרגול (אפשר לשאול לגבי שימוש ספציפי בפונקציות נוספות).

הימנעו מהכללה של ספריות אם אתם לא משתמשים בהן, וכמו כן הימנעו מהכללה של ספריות בקובץ ה-h, הספריות הנוספות אינן חלק מהממשק של המודול, ולמשתמש החיצוני לא אכפת איך ממומש הקוד בקובץ ה-c. כאשר יש צורך (ורק כאשר יש צורך) בהכללה של המודולים שאתם עצמכם כתבתם, יש לבצע זאת בעזרת מרכאות (כפי שמופיע בקובץ ex6.c ובקובץ sorting.h).
ניתן לממש פונקציות עזר, אין צורך להחצין אותן בקבצי ה-h. חשבו היטב באיזה מודול ראוי למקם כל אחת מהן.

הימנעו משימוש בקבועים בקוד. הכוונה היא גם לקבועים מספריים וגם לקבועי טקסט המשמשים להדפסה, את כולם ניתן להגדיר בעזרת const/define. אם יש קבוע שאתם משתמשים בו בכמה קבצים שונים, מקומו יהיה כמובן במודול utils, אין צורך להגדיר אותו יותר מפעם אחת.

הניחו כי כאשר אתם מבקשים מהמשתמש להכניס ערך בוליאני (0|1) אז הוא באמת יכניס ערך בוליאני.

הניחו כי כאשר אתם מבקשים מהמשתמש להכניס מספר מזהה (ID) אז הוא באמת יכניס ערך המתאים למשתנה מסוג int.

הניחו כי כאשר אתם מבקשים מהמשתמש להכניס שם של אדם או של אחד מילדיו, אז תוכנס מחרוזת תקינה באותיות אנגליות ללא רווחים (כלומר רצף תווים מהקבוצה [a-z,A-Z]) באורך שאינו עולה על 80 תווים. האורך כולל את תו הסיום \0, כלומר ניתן להגדיר את המערך שמשמש לקליטה זמנית של הנתון הזה בגודל 80, כמובן שמדובר בקבוע ולכן יש להגדיר אותו כפי שהוסבר קודם.

הניחו כי במידה ותתבקשו למיין את המעגל על פי קריטריון כלשהו, אז לא יופיעו במעגל אנשים עם ערך זהה של הקריטריון הזה.

נספח – Variadic Function

מצורפת דוגמה לפונקציה המקבלת מספר משתנה של פרמטרים ומחשבת את הממוצע שלהם. אין צורך להבין בדיוק מה קורה בכל שורה בפונקציה הזו, אלא רק איך להשתמש כראוי בספריה stdarg על מנת לחלץ את הפרמטרים שאינם ידועים.

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

double average(int count, ...) {
   double sum = 0;
   va_list ap;
   va_start(ap, count);
   for (int i = 0; i < count; i++) {
      sum += va_arg(ap, int);
   }
   va_end(ap);

   if (count == 0) {
      printf("0 arguments!\n");
      return 0;
   }
   return sum / count;
}
```

ניתן לקרוא לפונקציה average עם כמה פרמטרים שרוצים.

על מנת שהפונקציה תעבוד כראוי, יש להעביר לה את מספר הפרמטרים כפרמטר ראשון (זו לא חובה, הפונקציה printf למשל, סופרת כמה % מופיעים במחרוזת שמתקבלת כפרמטר).

לפרמטר זה חשיבות נוספת, משום שהוא הפרמטר האחרון של הפונקציה (לפני הפרמטרים הלא ידועים). בעזרתו הפונקציה תחשב את הכתובות של הפרמטרים הנוספים.

יש להכליל את הספריה stdarg (גם אצלנו יש להכליל אותה בקובץ person.c). כעת ניתן להצהיר על משתנה מסוג va_list.

הפקודה va_start(ap, count) מאתחלת את המשתנה ap בכתובת של הפרמטר הראשון הבלתי ידוע של הפונקציה, לפי סוף הכתובת של הפרמטר האחרון הידוע.

הפקודה va_arg מחלצת בכל קריאה משתנה בלתי ידוע נוסף. כאן הפונקציה average מניחה שכל הפרמטרים שהועברו לה הם מסוג int (זו לא חובה גם כן, הפונקציה printf מסיקה את סוג המשתנה לפי המחרוזת).

הפקודה va_end משחררת את המשתנה ap (בדומה לשחרור זיכרון בעזרת free).

דוגמה לפונקציית main המשתמשת בפונקציה:

```
int main() {
   printf("%f\n", average(4, 3, 1, -3, -1));
   printf("%f\n", average(3, 1, 2, 3));
   printf("%f\n", average(2, 3, 4));
   printf("%f\n", average(1, 2));
   printf("%f\n", average(0));
   return 0;
}
```

בתרגיל שלנו, עליכם להגדיר את הפונקציה CreateNext המקבלת שני פרמטרים ודאיים, כאשר השני הוא בוליאני ומסמן האם קיים פרמטר שלישי, מסוג *person.

Example 1

```
int main() {
    Person* head = InitTheHungerGame();
    PrintCircle(head);
    LetTheHungerGameBegin(head);
    return 0;
}
```

```
Add a person to the game? (0|1)
1
Name:
a
ID:
1
Num of kids:
0
Add a person to the game? (0|1)
1
Name:
b
ID:
2
Num of kids:
3
Kid #1 name:
p
Kid #2 name:
q
Kid #3 name:
r
Add a person to the game? (0|1)
1
Name:
c
ID:
3
Num of kids:
2
Kid #1 name:
q
Kid #2 name:
r
Add a person to the game? (0|1)
0
----------
Name: a
ID: 1
----------
Name: b
ID: 2
Kid #1: p
Kid #2: q
Kid #3: r
----------
Name: c
ID: 3
Kid #1: q
Kid #2: r
----------
a kills b and p and q and r
c kills a
Ah, ha, ha, ha, stayin' alive, stayin' alive! Ah, ha, ha, ha, "c" stayin' alive!
```

שימו לב שהדפסות kill והדפסת הניצחון מתבצעות בשורה אחת לכל הודעה למרות שזה נראה כאן מחולק.

Example 2

```
int main() {
    Person* head = InitTheHungerGame();
    PrintCircle(head);
    head = SortCircleByName(head);
    PrintCircle(head);
    head = SortCircleByID(head);
    PrintCircle(head);
    LetTheHungerGameBegin(head);
    return 0;
}
```

```
Add a person to the game? (0|1)
1
Name:
Nobody
ID:
1000
Num of kids:
0
Add a person to the game? (0|1)
1
Name:
Snape
ID:
37
Num of kids:
2
Kid #1 name:
p
Kid #2 name:
q
Add a person to the game? (0|1)
1
Name:
Harry
ID:
777
Num of kids:
3
Kid #1 name:
James
Kid #2 name:
Albus
Kid #3 name:
Lily
Add a person to the game? (0|1)
1
Name:
Dumbledore
ID:
100
Num of kids:
0
Add a person to the game? (0|1)
1
Name:
James
ID:
10
Num of kids:
1
Kid #1 name:
Harry
Add a person to the game? (0|1)
1
Name:
Voldemort
ID:
1
Num of kids:
9
Kid #1 name:
Barty
Kid #2 name:
Bellatrix
Kid #3 name:
Salazar
Kid #4 name:
Lucius
Kid #5 name:
Crabbe
Kid #6 name:
Goile
Kid #7 name:
Avery
Kid #8 name:
Nagini
Kid #9 name:
Horcruxes
Add a person to the game? (0|1)
0
----------
Name: Nobody
ID: 1000
```

Example 2

```
----------
Name: Snape
ID: 37
Kid #1: p
Kid #2: q
----------
Name: Harry
ID: 777
Kid #1: James
Kid #2: Albus
Kid #3: Lily
----------
Name: Dumbledore
ID: 100
----------
Name: James
ID: 10
Kid #1: Harry
----------
Name: Voldemort
ID: 1
Kid #1: Barty
Kid #2: Bellatrix
Kid #3: Salazar
Kid #4: Lucius
Kid #5: Crabbe
Kid #6: Goile
Kid #7: Avery
Kid #8: Nagini
Kid #9: Horcruxes
----------
----------
Name: Dumbledore
ID: 100
----------
Name: Harry
ID: 777
Kid #1: James
Kid #2: Albus
Kid #3: Lily
----------
Name: James
ID: 10
Kid #1: Harry
----------
Name: Nobody
ID: 1000
----------
Name: Snape
ID: 37
Kid #1: p
Kid #2: q
----------
Name: Voldemort
ID: 1
Kid #1: Barty
Kid #2: Bellatrix
Kid #3: Salazar
Kid #4: Lucius
Kid #5: Crabbe
Kid #6: Goile
Kid #7: Avery
Kid #8: Nagini
Kid #9: Horcruxes
----------
----------
Name: Voldemort
ID: 1
Kid #1: Barty
Kid #2: Bellatrix
Kid #3: Salazar
Kid #4: Lucius
Kid #5: Crabbe
Kid #6: Goile
Kid #7: Avery
Kid #8: Nagini
Kid #9: Horcruxes
----------
Name: James
ID: 10
Kid #1: Harry
----------
Name: Snape
ID: 37
Kid #1: p
Kid #2: q
----------
Name: Dumbledore
ID: 100
----------
Name: Harry
ID: 777
Kid #1: James
Kid #2: Albus
Kid #3: Lily
----------
```

Example 2

Name: Nobody
ID: 1000

Voldemort kills James and Harry
Snape kills Dumbledore
Harry kills Nobody
Voldemort kills Snape and p and q
Harry kills Voldemort and Barty and Bellatrix and Salazar and Lucius and Crabbe and Goile and Avery and Nagini and Horcruxes
Ah, ha, ha, ha, stayin' alive, stayin' alive! Ah, ha, ha, ha, "Harry" stayin' alive!

שימו לב שהדפסות kill והדפסת הניצחון מתבצעות בשורה אחת לכל הודעה למרות שזה נראה כאן מחולק.

Example 3

```
int main() {
    Person* head = InitTheHungerGame();
    PrintCircle(head);
    InsertLaters(head);
    PrintCircle(head);
    head = RemoveCowards(head);
    PrintCircle(head);
    LetTheHungerGameBegin(head);
    return 0;
}
```

```
Add a person to the game? (0|1)
1
Name:
a
ID:
1
Num of kids:
0
Add a person to the game? (0|1)
1
Name:
b
ID:
2
Num of kids:
0
Add a person to the game? (0|1)
1
Name:
c
ID:
3
Num of kids:
0
Add a person to the game? (0|1)
0
----------
Name: a
ID: 1
----------
Name: b
ID: 2
----------
Name: c
ID: 3
----------
Are you late? (0|1)
1
Insert your best friend's ID:
1
Name:
d
ID:
4
Num of kids:
0
Are you late? (0|1)
1
Insert your best friend's ID:
8
No Such ID: 8
Are you late? (0|1)
1
Insert your best friend's ID:
3
Name:
e
ID:
5
Num of kids:
0
Are you late? (0|1)
1
Insert your best friend's ID:
4
Name:
f
ID:
6
Num of kids:
0
Are you late? (0|1)
0
----------
Name: a
ID: 1
----------
Name: d
ID: 4
----------
Name: f
ID: 6
----------
Name: b
```

Example 3

```
ID: 2
----------
Name: c
ID: 3
----------
Name: e
ID: 5
----------
Is there a coward in here? (0|1)
1
Insert your ID, you, lame pudding-heart coward:
2
Let the world know that you are a coward, b!
Is there a coward in here? (0|1)
1
Insert your ID, you, lame pudding-heart coward:
1
Let the world know that you are a coward, a!
Is there a coward in here? (0|1)
1
Insert your ID, you, lame pudding-heart coward:
3
Let the world know that you are a coward, c!
Is there a coward in here? (0|1)
0
----------
Name: d
ID: 4
----------
Name: f
ID: 6
----------
Name: e
ID: 5
----------
d kills f
e kills d
Ah, ha, ha, ha, stayin' alive, stayin' alive! Ah, ha, ha, ha, "e" stayin' alive!
```

שימו לב שהדפסות kill והדפסת הניצחון מתבצעות בשורה אחת לכל הודעה למרות שזה נראה כאן מחולק.

Example 4

```
int main() {
    Person* head = InitTheHungerGame();
    PrintCircle(head);
    InsertLaters(head);
    PrintCircle(head);
    head = RemoveCowards(head);
    PrintCircle(head);
    head = SortCircleByName(head);
    PrintCircle(head);
    head = SortCircleByID(head);
    PrintCircle(head);
    LetTheHungerGameBegin(head);
    return 0;
}
```

```
Add a person to the game? (0|1)
1
Name:
Coward
ID:
10
Num of kids:
1
Kid #1 name:
CowardToo
Add a person to the game? (0|1)
0
----------
Name: Coward
ID: 10
Kid #1: CowardToo
----------
Are you late? (0|1)
0
----------
Name: Coward
ID: 10
Kid #1: CowardToo
----------
Is there a coward in here? (0|1)
1
Insert your ID, you, lame pudding-heart coward:
10
Let the world know that you are a coward,
Coward!
```

Example 5

```
int main() {
    Person* head = InitTheHungerGame();
    PrintCircle(head);
    InsertLaters(head);
    PrintCircle(head);
    head = RemoveCowards(head);
    PrintCircle(head);
    head = SortCircleByName(head);
    PrintCircle(head);
    head = SortCircleByID(head);
    PrintCircle(head);
    LetTheHungerGameBegin(head);
    return 0;
}
```

```
Add a person to the game? (0|1)
0
```

שימו לב שהדפסות Let the world know
מתבצעת בשורה אחת לכל הודעה למרות שזה
נראה כאן מחולק.