מבוא לתכנות מערכות (234122)

סמסטר אביב 2015

פתרון מבחן מסכם מועד א', 24 יולי 2015

משך המבחן: 3 שעות.
חומר עזר: ניתן להשתמש בכל חומר עזר מודפס. חוברת עזר מצורפת לבחינה.

הנחיות והוראות:

- מלאו את הפרטים בדף השער המצורף.
- בדקו שיש 26 עמודים (4 שאלות) במבחן, כולל עמוד זה.
- **כתבו את התשובות על טופס המבחן בלבד**, במקומות המיועדים לכך. שימו לב שהמקום המיועד לתשובה אינו מעיד בהכרח על אורך התשובה הנכונה.
- השתמשו בעמודים הריקים בבחינה כדפי טיוטה וכן לכתיבת תשובותיכם. סמנו טיוטות באופן ברור על מנת שהן לא תיבדקנה.
- **ניתן בהחלט להשתמש בעיפרון ומחק** (על אף מה שכתוב בדף השער), פרט לדף השער עצמו אותו יש למלא בעט.
- **יש להקפיד על כתיבה ברורה ומסודרת של התשובות** ועל קוד קריא ובהיר.
- **אין צורך להגדיר define#** לקבועים המופיעים בקוד שלכם.
- אין צורך לתעד את הקוד או להקפיד על code conventions כלשהם.

צוות הקורס 234122
מרצים: דר' יחיאל קמחי, דר' רן רובינשטיין (מרצה אחראי)
מתרגלים: עופר גבעולי, שגיא לאופר, עידן שוורץ, יורי פלדמן, תומר גולני, שון הדר,דני רסין (מתרגל אחראי)

בהצלחה!

שאלה 1: שפת C (20 נק')

סעיף א (10 נק')

בשאלה זו נתון לכם קוד C שידוע שמתקמפל בהצלחה, אך מכיל מספר שגיאות מהותיות. עליכם לקרוא את הקוד, לזהות את השגיאות שבו, ולמלא בקצרה את הטבלה בהמשך (טעות אחת בכל שורה בטבלה). שימו לב שאין צורך להבין את מטרת הקוד – השגיאות הן שגיאות בסיסיות בשפת C שעשויות לגרום לתוכנית להתנהג באופן לא רצוי או לא מוגדר.

עליכם למצוא 5 טעויות. לנוחותכם – מספרי השורות מופיעים משמאל לקוד, התייחסו למספרי שורות אלה בעמודה הראשונה בטבלה. בעמודה האחרונה של הטבלה כתבו בקצרה כיצד יש לשנות את הקוד על מנת לתקן את הטעות (כתבו את קטע הקוד המתוקן והיכן הוא צריך להופיע).

```
1   #define SOME_OPERATION(a,b) a+b
2
3   typedef struct {
4       char* name;
5       int nameLen;
6       int val;
7   } SomeType;
8
9   SomeType* f(int x, int y, int z, char* name)
10  {
11      SomeType a;
12
13      assert(name != NULL);
14      a.name = malloc(strlen(name) + 1);
15      strcpy(a.name, name);
16      assert( (a.nameLen = strlen(name)) > 0 );
17
18      a.val = SOME_OPERATION(x,y) * z;
19      if (a.val < 0) {
20        return NULL;
21      }
22
23      return &a;
24  }
```

שורה	מהי הטעות?	כיצד לתקן את הטעות?
1 (ו/או 18)	חסרים סוגריים בהגדרת ה-#define	צריך להגדיר את ה-#define כך: ((a)+(b))
14	אין בדיקת ערך החזרה של malloc	יש לבדוק ש-malloc לא מחזיר NULL
16	ביצוע קוד הכרחי לתוכנית בתוך assert	a.nameLen = strlen(name); assert(a.nameLen > 0);
20	זליגת זיכרון	יש לעשות free ל-a.name לפני היציאה מ-f
23	החזרת מצביע למשתנה מקומי	יש לבצע malloc ל-a או להחזיר by value

התקבל גם, למרות שלא נכלל ברשימה הנ"ל:

9	name מצביע קלט שלא מוגדר כקבוע	צריך להגדיר את name כ-const char*

סעיף ב (10 נקודות)

ממשו ב-C פונקציה גנרית בשם count_if אשר מקבלת מערך של איברים מטיפוס כלשהו, וסופרת כמה מהאיברים במערך זה מקיימים תנאי כלשהו. יש לאפשר למשתמש לבחור את התנאי כרצונו.

לדוגמה, עבור המערך הבא של מספרים שלמים:

```
int a[6] = { 1, -2, 3, 4, -8, 7 };
```

הפונקציה תחזיר 2 אם הקריטריון לספירה הוא שאיבר הוא קטן מ-0 (ישנם 2 מספרים במערך שקטנים מ-0), והיא תחזיר 3 אם הקריטריון לספירה הוא שאיבר הוא זוגי (יש 3 מספרים זוגיים במערך).

1. כתבו את מימוש הפונקציה count_if. ניתן להניח שהארגומנטים שהפונקציה מקבלת תקינים. שימו לב שהמימוש שלכם צריך לאפשר הפעלה (למשל) על המערך a למעלה.
2. הראו כיצד ניתן להשתמש בפונקציה שכתבתם על מנת לספור את מספר האיברים הזוגיים במערך a הנ"ל.

```
typedef bool (*Cond)(void* x, void* param);  // אופציונאלי

int count_if(void* a, int n, int sz, Cond f, void* param) {
    int count = 0;
    for (int i=0; i<n; ++i) {
        void* elem = ((char*)a) + i*sz;
        count += f(elem, param);
    }
    return count;
}

// הערה: מימוש של חלוקה ב-2 בלבד גם מתקבלת
bool dividesBy(void* elem, void* param) {
    return *(int*)elem % *(int*)param == 0;
}

int val = 2;
int count = count_if(a, 6, sizeof(a[0]), dividesBy, &val);
```

שאלה 2: C ADT (32 נק')

בשאלה זו נניח שיש לנו אתר חברתי המאחסן אוסף של משתמשים, ולכל משתמש אוסף של תחביבים. מטרת השאלה היא לבנות מערכת פשוטה שתמצא משתמשים בעלי תחומי עניין משותפים.

לצורך השאלה נתון לכם ממשק ADT בסיסי המייצג משתמש במערכת, ונקרא User (מופיע בסוף השאלה). כמו כן נתונות לשימושכם פונקציות העזר הבאות:

1. פונקציות לטיפול במחרוזות:

```
void* strCopy(void* str);
void strFree(void* str);
bool strIsEqual(void* str1, void* str2);
```

2. פונקציות להעתקה ופינוי של User:

```
void* userCopy(void* user);
void userFree(void* user);
```

הערות כלליות לשאלה:

1. הניחו ששמות המשתמשים במערכת ייחודיים (כלומר, אין שני משתמשים בעלי אותו שם במערכת).
2. שימו לב שבשאלה זו נעשה שימוש ב-ADT קיימים ואין צורך לממש ADT חדש.
3. לאורך כל השאלה, אין צורך לטפל בשגיאות של פונקציות ה-ADT שאתם עושים בהן שימוש, וניתן להניח שהן מצליחות. כמו כן ניתן להניח שפרמטרים מטיפוס מצביע אינם NULL, ואין צורך לבדוק זאת. יש לטפל בשגיאות אחרות כרגיל.

סעיף א' (3 נקודות)

כתבו פונקציה המקבלת כפרמטרים שני משתמשים מטיפוס User, ומספר שלם n, ובודקת האם לשני המשתמשים יש לפחות n תחביבים משותפים. הפונקציה מחזירה ערך מתאים מטיפוס bool.

סעיף ב' (7 נקודות)

כתבו פונקציה המקבלת כפרמטרים שם של משתמש כלשהו במערכת (מטיפוס *char), מספר שלם n, וקבוצה של כל משתמשי המערכת (Set של טיפוס User). על הפונקציה להחזיר קבוצה של *char המכילה את שמות כל המשתמשים שיש להם לפחות n תחביבים משותפים עם המשתמש הנתון (אין לכלול בקבוצה את שם המשתמש שהועבר כפרמטר). אם שם המשתמש הנתון לא נמצא בקבוצה, החזירו קבוצה ריקה.

לנוחותכם נתונה לכם פונקציית העזר הבאה (גם לסעיפים הבאים) המקבלת קבוצה של User ושם משתמש, ומחזירה מצביע למשתמש עם שם זה (לא עותק), או NULL אם אין משתמש עם שם זה בקבוצה:

```
User findUser(Set users, char* name);
```

סעיף ג' (12 נקודות)

כתבו פונקציה אשר מקבלת שם של קובץ קלט, קוראת מתוכו את רשימת כל המשתמשים במערכת ואת התחביבים שלהם, ומחזירה קבוצה המכילה את כל משתמשי המערכת. על הפונקציה ליצור אובייקט מטיפוס User עבור כל משתמש, ולשמור את האובייקטים הללו בקבוצה. יש לממש גם כל פונקציית עזר שנדרשת לצורך פתרון סעיף זה.

קובץ הקלט הינו קובץ טקסט פשוט המכיל מספר שורות, כאשר כל שורה הינה בפורמט הבא:

```
<name> <hobby>
```

כלומר, שם המשתמש, לאחריו רווח אחד או יותר, ולאחריו שם של תחביב. ניתן להניח שאין רווחים בשם המשתמש או בשם התחביב, ושכל אחד מהם באורך 99 תווים לכל היותר. שימו לב שאותו שם משתמש עשוי להופיע מספר פעמים בקובץ, כמספר התחביבים של אותו המשתמש. לדוגמה, קובץ הקלט עשוי להראות כך:

```
Tomer Basketball
Noa Reading
Noa Tennis
Ran Reading
Ran Television
Ran Tennis
Tomer Reading
Tomer Tennis
Meital Basketball
```

תזכורת:
קריאת מחרוזת מהקלט באמצעות תג הבקרה %s מדלגת על whitespace (רווחים וירידות שורה), וקוראת מילה אחת בדיוק ללא whitespace.

סעיף ד' (5 נקודות)

כתבו פונקציית main קצרה לתוכנית. הפונקציה תקבל שני פרמטרים בשורת הפקודה: שם קובץ קלט המכיל את המידע על משתמשי המערכת ותחביביהם, ושם של משתמש. הפונקציה תקרא את קובץ הקלט באמצעות הפונקציה מסעיף ג', ולאחר מכן תיקרא לפונקציה מסעיף ב' ותמצא את כל המשתמשים במערכת שיש להם לפחות 5 תחביבים משותפים עם המשתמש הנתון. התוכנית תדפיס את מספר המשתמשים שמצאה, או 0 אם המשתמש המבוקש לא קיים במערכת. ניתן להניח שהועברו בדיוק 2 פרמטרים ל-main.

סעיף ה' (5 נקודות)

כפי שראינו, בשאלה זו בחרנו לייצג את הנתונים אודות המשתמשים באמצעות Set של User, כאשר כל User מאחסן את שם המשתמש שלו (מטיפוס *char) וכן את אוסף התחביבים (Set של מחרוזות). עם זאת, ניתן היה לחשוב על אופציות אחרות לייצוג המידע.

עבור כל אחת מן האופציות הבאות, ציינו סיבה אחת בולטת מדוע היא פחות טובה מהייצוג שהשתמשנו בו בשאלה. ניקוד גבוה יותר יינתן לסיבות הקשורות בתכן של הקוד (למשל, מדוע מבני הנתונים המוצעים יתקשו יותר להתמודד עם דרישות עתידיות מהתוכנית).

1. ייצוג המשתמשים באמצעות Map, כאשר המפה מקשרת בין שם המשתמש מטיפוס *char (ה-key של המפה) לבין קבוצת התחביבים של המשתמש (מטיפוס Set של מחרוזות).

2. ייצוג המשתמשים שוב באמצעות Map, אך הפעם המפה מקשרת בין טיפוס User המכיל את כל פרטי המשתמש, לא כולל התחביבים שלו, לבין קבוצת התחביבים של המשתמש (מטיפוס Set של מחרוזות).

נספח: קובץ הממשק User.h

```
#ifndef USER_H_
#define USER_H_
#include "set.h"
#include <stdbool.h>

typedef struct user_t* User;

// מייצר משתמש חדש עם השם הנתון
User userCreate(char* name);

// מפנה משתמש
void userDestroy(User user);

// מייצר עותק של משתמש
User userCopy(User user);

// מוסיף תחביב למשתמש נתון
// מחזיר true בהצלחה, או false במקרה של שגיאת זיכרון
bool userAddHobby(User user, char* hobby);

// מחזיר קבוצת מחרוזות (מטיפוס char*) המכילה את התחביבים של המשתמש
// פונקציה זו אינה יכולה להיכשל
// חשוב: אין לשנות או להרוס את ה-Set המוחזר
Set userGetHobbies(User user);

// מחזיר את השם של המשתמש
// פונקציה זו אינה יכולה להיכשל
// חשוב: אין לשנות או להרוס את המחרוזת המוחזרת
char* userGetName(User user);

#endif
```

סעיף א:

```
bool haveCommonHobbies(User u1, User u2, int n) {
  Set s1 = userGetHobbies(u1);
  Set s2 = userGetHobbies(u2);
  Set common = setIntersection(s1, s2);
  bool res = setGetSize(common) >= n;
  setDestroy(common);
  return res;
}
```

סעיף ב:

```
Set usersWithCommonHobbies(Set users, char* name, int n) {
  Set res = setCreate(strCopy, strFree, strIsEqual);
  User u = findUser(users, name);
  if (u == NULL) {
    return res;
  }
  SET_FOREACH(User, user, users) {
    char* username = userGetName(user);
    if (strcmp(username, name) != 0 &&
        haveCommonHobbies(u, user, n)
    {
      setAdd(res, username);
    }
  }
  return res;
}
```

סעיף ג:

```
Set readUserFile(char* filename) {
  FILE* f = fopen(filename, "r");
  if (!f) {
    return NULL;
  }
  Set users = setCreate(userCopy, userFree, userIsEqual);
  char name[100], hobby[100];
  while (fscanf(f, "%s%s", name, hobby) == 2) {
    addUserHobbyPair(users, name, hobby);
  }
  fclose(f);
  return users;
}

void addUserHobbyPair(Set users, char* name, char* hobby) {
  User u = findUser(users, name);
  if (u == NULL) {
    u = userCreate(name);
    userAddHobby(u, hobby);      // הערה: למה זה שגוי לבצע
    setAdd(users, u);            // פעולות אלה בסדר הפוך?
    userDestroy(u);
  }
  else {
    userAddHobby(u, hobby);
  }
}

bool userIsEqual(void* u1, void* u2) {
  char* name1 = userGetName((User)u1);
  char* name2 = userGetName((User)u2);
  return strcmp(name1,name2) == 0
}
```

סעיף ד:

```
int main(int argc, char* argv[]) {
  if (argc != 3) {
    return 1;
  }
  char* filename = argv[1];
  char* username = argv[2];
  Set users = readUserFile(filename);
  if (users == NULL) {
    return 1;
  }
  Set s = usersWithCommonHobbies(users, username, 5);
  printf("Number of users = %d\n", setGetSize(s));
  setDestroy(s);
  setDestroy(users);
  return 0;
}
```

סעיף ה:

1. באופציה זו אין כלל טיפוס המייצג משתמש, ולכן לא ניתן להוסיף בקלות תכונות או פעולות עבור משתמש, או להעביר משתמש לפונקציה.

2. באופציה זו התחביבים מנותקים מיתר התכונות של משתמש, והשליטה על רשימת התחביבים של המשתמש אינה בידיים שלו. פונקציונליות שיהיה קשה להוסיף, למשל, הינה הגבלה על אילו תחביבים אפשר להוסיף למשתמש מסוים (אולי משתמש אוהב רק תחביבים שקשורים בספורט?), או עדכון פרמטרים נוספים של המשתמש כאשר רשימת התחביבים שלו משתנה (למשל, אולי למשתמש יש פרמטר "זמן פנוי" שמתעדכן על פי מספר התחביבים שלו?).

שאלה 3: Bash (15 נק')

בשאלה זו נבנה תסריט שיעזור למיין קבצי מוסיקה בספריות, על פי זמר ואלבום. לשם הפשטות, אנו לא ננסה לקרוא מידע מתוך קבצי mp3, אלא נניח שמדובר בקבצי טקסט פשוטים.

לפני תחילת ריצת התסריט, אנו נניח שקיימת תיקייה ראשית יחידה ובה קבצים עם סיומת **song.**, כאשר כל קובץ הוא בפורמט הבא:

```
<artist>,<album>
```

כלומר, שורה אחת בלבד ובה שם האמן ושם האלבום מופרדים בפסיק, בלי רווחים סביב הפסיק.

בתום ריצת התסריט נרצה שהספרייה הראשית תכיל אך ורק אוסף תיקיות ששמן כשם האמנים, ובכל תיקייה כזו אוסף תתי-תיקיות ששמותיהם כשמות האלבומים של אותו אומן. בתוך תתי-תיקיות אלו יימצאו קבצי ה-song., שהוזזו מהספרייה הראשית לספריה המתאימה לכל אחד ע"פ האומן והאלבום שלו. לדוגמה, נניח שהתיקייה הראשית מכילה את קבצי השירים הבאים:

```
poison.song:          alice_cooper,trash
snakebite.song:       alice_cooper,hey_stoopid
hells_bells.song:     acdc,back_in_black
```

בתום ריצת התסריט תהיה בספרייה הראשית תיקייה אחת בשם "alice_cooper" שתכיל 2 תתי-תיקיות – "trash" המכילה את הקובץ poison.song, ו-"hey_stoopid" המכילה את snakebite.song. כמו כן תהיה בתיקייה הראשית תיקייה נוספת בשם "acdc" שתכיל תת-תיקייה בשם "back_in_black", ובה הקובץ hells_bells.song.

אנו נבנה את התסריט הראשי בשלבים:

א. כתבו תסריט שכשמורץ מתוך התיקייה הראשית מזיז את כל קבצי השירים לתיקיות עם שמות האמנים המתאימים להם. על התסריט לייצר את הספריות הנדרשות.

ב. כתבו תסריט שכשמורץ מהתיקייה הראשית מזיז את השירים מספריות האומנים לתתי-תיקיות עם שמות האלבומים הרלוונטיים. הניחו כי תסריט א' הורץ לפני הקריאה לתסריט זה. על התסריט לייצר את תתי-התיקיות הנדרשות.

ג. כתבו את התסריט הראשי המריץ את שני התסריטים הקודמים, ומבצע את פעולת ההזזה המלאה של השירים. התסריט מורץ אף הוא מהתיקייה הראשית.

הערות לשאלה:

- כל השמות בשאלה (שירים, אמנים ואלבומים) מכילים אותיות אנגליות קטנות וקווים תחתונים ('_') בלבד.
- התסריטים מסעיפים א' ו-ב' אינם מורצים באופן ישיר, אלא רק ע"י תסריט ג'.
- ניתן להניח כי כשמורץ התסריט מסעיף ג', התיקייה הראשית אינה מכילה תיקיות כלשהן.
- כל קבצי ה-song. במבנה תקין ואין צורך לבדוק זאת.
- התיקייה הראשית עשויה להכיל קבצים נוספים מלבד קבצי שירים (למשל, את קבצי התסריטים), אך ניתן להניח כי אין אף קובץ עם שם של להקה או אלבום.
- מותר ומומלץ להשתמש בפונקציות עזר.

.1

```
#!/bin/bash
function getArtist {
  read line < "$1"
  echo $line | cut -d"," -f1
}

for file in *.song ; do
  artist=`getArtist "$file"`
  if [[ ! (-d "$artist")]] ; then
    mkdir "$artist"
  fi
  mv "$file" "$artist"
done
```

.2

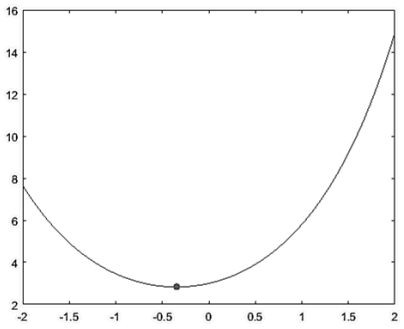
```
#!/bin/bash
function getAlbum {
  read line < "$1"
  echo $line | cut -d"," -f2
}

for artist in * ; do
  if [[  (-d "$artist") ]] ; then
    cd "$artist"
    for file in *.song ; do
      album=`getAlbum "$file"`
      if [[ ! (-d "$album")]] ; then
        mkdir "$album"
      fi
      mv "$file" "$album"
    done
    cd ..
  fi
done
```

.3

```
#!/bin/bash
./createArtistFolder
./createAlbumFolder
```

שאלה 4: C++ (33 נק')

בשאלה זו נממש אלגוריתם שמטרתו למצוא את המינימום של פונקציה $f(x)$. נזכיר שעבור פונקציה כלשהי $f(x)$, **המינימום שלה** הינו הנקודה x שבה $f(x)$ קטן ביותר. לדוגמה, בפונקציה למטה, המינימום של הפונקציה הוא בנקודה $x = -0.35$, ומסומן בנקודה השחורה בשרטוט:



אנו נממש את האלגוריתם הבא, הידוע בשם steepest descent, ומובא כאן בפסאודו-קוד. שימו לב שהאלגוריתם למעשה אינו דורש לחשב את $f(x)$, אלא רק את הנגזרת שלה $f'(x)$. לשם פשטות נסמן את פונקציית הנגזרת ב- $g(x) = f'(x)$.

<u>קלט</u>: פונקציית הנגזרת $g(x)$, ניחוש התחלתי x_0, פרמטר α

<u>אלגוריתם</u>:

1. אתחל $x = x_0$
2. בצע לכל היותר 1000 פעם:
   א. $d = g(x)$
   ב. אם $|d| < 10^{-8}$ (הערך המוחלט של d מאוד קטן), <u>עצור</u> והחזר את x הנוכחי
   ג. $x = x - \alpha \cdot d$
3. אם הגעת לפה, החזר שגיאה – האלגוריתם כשל

בשאלה זו נרצה לכתוב פונקציה בשם descent() המממשת את האלגוריתם הנ"ל. הפונקציה descent() תקבל שלושה פרמטרים: פרמטר g המאפשר לחשב את הנגזרת $f'(x)$, ניחוש התחלתי x_0, וערך ממשי α. הפונקציה מחזירה את המינימום x שמצאה.

<u>סעיף א (6 נקודות)</u>

נרצה לממש את descent() באופן כללי ככל האפשר, כלומר כך שהיא תאפשר למשתמש לחשב את המינימום של **כל פונקציה $f(x)$ שיבחר**. כפי שראינו בקורס, יש ב-C++ שתי צורות שבהן ניתן לממש כלליות כזו: באמצעות תבניות (templates) ושימוש ב-function objects, או באמצעות ירושה (inheritance).

כתבו בקצרה יתרון אחד של כל אחת מן הגישות הללו לעומת הגישה השנייה (למשל, יתרונות עבור המשתמש ב-descent()).

סעיף ב (14 נקודות)

ממשו את הפונקציה ()descent תוך שימוש בירושה כדי לענות על דרישת הכלליות. לשם כך הגדירו מחלקה אבסטרקטית בשם Function המייצגת פונקציה מתמטית כללית, והשתמשו בה במימוש של descent.

הראו כיצד ניתן להשתמש בפונקציה שכתבתם לצורך חישוב המינימום של הפונקציה $f(x) = 2e^{x} + e^{-x}$ (הפונקציה המשורטטת בעמוד הקודם). עבור $f(x)$ הזו, על הקריאה ל-()descent להראות כך:

```
x = descent(ExpDerivFunc(2,1), 0.1, 3.0);  // alpha = 0.1, x0 = 3.0
```

שימו לב שהפרמטרים 2 ו-1 בקוד הם המקדמים של e^{x} ו-e^{-x}, בהתאמה, ושהקריאה ()ExpDerivFunc הינה קריאה לבנאי של מחלקה המייצגת את הנגזרת של פונקציות מהצורה $ae^{x} + be^{-x}$. הפונקציה ()descent עצמה היא פונקציה עצמאית ואינה שייכת למחלקה כלשהי.

הערות:

1. אין להשתמש בתבניות (templates) או בהעמסת אופרטורים בסעיף זה.
2. במקרה של שגיאה (שלב 3 באלגוריתם), זרקו את החריגה הבאה:

```
class DidNotConverge : public std::exception { };
```

סעיף ג (13 נקודות)

בסעיף זה נממש "קוד מתאם" שיאפשר שימוש בקוד קיים עם הפונקציה ()descent שמומשה בסעיף ב'.

אנו נניח שלמשתמש של ()descent כבר יש אוסף פונקציות C קיימות, ואולי גם אובייקטים שונים המספקים אופרטור סוגריים (function objects), ומחשבים את פונקציית הנגזרת $g(x)$ עבור פונקציות $f(x)$ שונות. המשותף לכל אלו הוא שניתן לקרוא להם באמצעות הסינטקס y=foo(x) עבור x ו-y מטיפוס double, כאשר foo הוא או פונקציית C רגילה, או אובייקט המספק אופרטור סוגריים. הקריאה foo(x) מחזירה הנגזרת של פונקציה כלשהי בנקודה x.

כתבו קוד שיאפשר שימוש בפונקציה ()descent מסעיף ב' עם כל אובייקט שיש לו אופרטור סוגריים כנ"ל. בפרט, עבור פונקציית C כלשהי עם החתימה double deriv(double), על הקוד לאפשר קריאה ל-()descent באמצעות הסינטקס הבא:

```
x = descent(funcAdapter(deriv), 0.1, 3.0);
```

כמו כן אם מחלקה כלשהי Deriv מספקת אופרטור סוגריים המחשב את פונקציית הנגזרת, על הקוד לאפשר קריאה ל-()descent באמצעות הסינטקס הבא:

```
x = descent(funcAdapter(Deriv(…)), 0.1, 3.0);
```

כאשר (...) מייצג את הפרמטרים של הבנאי של Deriv (אם ישנם).

הערות:

1. בסעיף זה ניתן להשתמש בתבניות (templates).
2. רמז לפתרון: ()funcAdapter בקוד למעלה הינה פונקציה. שימו לב שהיא יכולה לקבל ארגומנטים מטיפוסים שונים. הפתרון הנכון לסעיף משלב הגדרה של מחלקה חדשה, וכן הגדרה של הפונקציה ()funcAdapter המחזירה אובייקט ממחלקה זו.

סעיף א:

1. יתרונות של תבנית: שימוש בתבניות יאפשר למשתמש של descent להעביר לה כל פונקציית C קיימת שיש לה אופרטור סוגריים, וכל אובייקט שיש לו אופרטור סוגריים, ללא צורך בכתיבת קוד נוסף (בפרט, אין צורך לעטוף את הפונקציה הקיימת עם מחלקה היורשת ממחלקת הבסיס).

2. יתרונות של ירושה: שימוש בירושה יאפשר לקבוע את זהות הפונקציה שמועברת ל-descent בזמן ריצה ולא רק בזמן קומפילציה, וכך משפרת משמעותית את השימושיות שלה (כזכור, עבור תבניות הקומפיילר צריך לדעת כבר בזמן קומפילציה את הטיפוס שמועבר לתבנית, לעומת ירושה בה הטיפוס מתברר בזמן ריצה).

סעיף ב:

```
class Function
{
public:
    virtual double compute(double x) const = 0;
    virtual ~Function() {}
};

double descent(const Function& deriv, double alpha,
               double x0 = 0.0)
{
  double x = x0;
  for (int i=0; i<1000; ++i) {
      double d = deriv.compute(x);
      if (fabs(d) < 1e-8) return x;
      x = x-alpha*d;
  }
  throw DidNotConverge();
}

class ExpDerivFunc : public Function
{
  double a, b;
public:
  ExpDerivFunc(double a, double b) : a(a), b(b) {}
  double compute(double x) const override {
    return a*exp(x) - b*exp(-x);
  }
};
```

סעיף ג:

```
template <class Func>
class FunctionAdapter : public Function
{
  Func f;
public:
  FunctionAdapter(Func f) : f(f) {}
  double compute(double x) const override {
    return f(x);
  }
};

template <class Func>
FunctionAdapter<Func> adaptFunction(Func f)
{
    return FunctionAdapter<Func>(f);
}
```