מבוא לתכנות מערכות - חורף תשע"ז

הנחיות להכנת תרגילי הבית

בקורס יהיו 4 תרגילי בית גדולים שעליהם קיימת חובת הגשה.

המשקל של תרגיל הבית הראשון בציון הסופי יהיה נמוך, כדי לתת לכם הזמנות להסתגל לדרישות וללמוד מטעויות בלי שהציון הסופי ייפגע כמעט. משקל התרגילים הבאים יהיה גדול יותר.

הגשת תרגילי הבית תהיה בזוגות בלבד. אתם מוזמנים להשתמש בפורום הסטודנטים במודל כדי לפרסם מודעות "מחפש שותף". במקרים נדירים ביותר תאושר הגשת תרגילי הבית לבד. בשום מקרה לא תאושר הגשת תרגילי הבית בקבוצות גדולות משתיים.

מותר ומומלץ לדון בתרגילים עם זוגות אחרים בעל-פה בלבד וללא התבוננות בקוד של אחרים. זכרו: "מותר לחלוק רעיונו ... אסור לחלוק קוד!".

העתקות ייענשו בחומרה רבה.

בדיקת תרגילי הבית

בדיקת תרגילי הבית תעשה בשתי דרכים:

1. בדיקה יבשה, במסגרתה נקרא את הקוד שלכם וננקד לפי האיכות שלו, בהתאם לעקרונות התכנות הנכון שלמדתם בתרגול.
2. בדיקה רטובה, המכונה גם Black Box Testing.
הבדיקה הרטובה תבוצע על ידי מערכת בדיקה אוטומטית. המערכת תקמפל את הקוד שלכם, ואז תריץ מספר טסטים. בכל טסט, התכנית שלכם תקבל סט שונה של קלטים, ותוציא פלט. אם התוכנית שלכם מוציאה פלט שזהה בדיוק לפלט הצפוי לפי ההנחיות בתרגיל הבית, תקבלו ציון עובר בטסט. אחרת, תקבלו ציון נכשל בטסט.

ציוני התרגיל יינתנו לפי:

1. הצלחת הקומפילציה, ללא שגיאות (ERRORS) ועם מינימום אזהרות (Warnings).
2. בדיקה רטובה: הצלחת ייצור קובץ פלט נכון למספ' טסטים שונים
3. הערכת הבודק לכתיבה נכונה של הקוד
4. הערכת הבודק לתיעוד נכון

איחורים בהגשה

יש לפנות לבודקי התרגילים לגבי איחורים הנובעים מסיבות מוצדקות (מילואים, אשפוז וכו').
כל יום איחור יוביל להורדה של 5 נקודות מציון התרגיל. בכל מקרה, לא יתקבלו תרגילים באיחור העולה על שבוע.

איך מגישים

ההגשה תתבצע באמצעות תוכנת DROPBOX. כל זוג יפתח תיקיה בשם 2ID1_ID כאשר כל ID מכיל תשע ספרות. במידה ויש סטודנט יחיד המגיש תרגילים הוא יפתח תיקיה בשם 000000000_ID.

יש לעשות שיתוף של תיקיית ה-DROPBOX עם ispchecker@gmail.com.

בתוך תיקיית השיתוף יגישו הסטודנטים את התרגילים, כל תרגיל בתיקייה נפרדת ששמה ex#.

לדוגמא:

Dropbox\222222222_111111111\ex2

הקבצים המוגשים בתיקייה יהיו כל קבצי הפרוייקט עליו עבדתם, כולל הקובץ המגדיר את הפרויקט בסביבת העבודה Microsoft Visual Studio, אשר הסיומת שלו היא vcxproj.

(רמז כדי לבדוק אם ההגשה שלכם נכונה : נסו לקמפל את הפרויקט שלכם ב command line ממיקום הROOT של תיקיית הגשת התרגיל - כמו בדוגמא).

בנוסף לקבצי הקוד, **יש להגיש (בתיקיית האם של התרגיל) קובץ Code מסוג .doc/docx אשר בו כתוב כל הקוד של כל קבצי ה.c וה-.h שקידדתם בעצמכם.**

הערות חשובות:

1. יש לשתף את התיקייה אשר שמה הוא 2ID1_ID עם חשבון בדיקת התרגילים. אין לשלוח לינק לתיקייה ב**DROPBOX** לחשבון המייל הנ"ל. פעולה זו תחשב כחוסר הגשה של התרגילים, והם לא ייבדקו.

2. יש לוודא שהבודקים אישרו את שיתוף התיקייה (מופיע בממשק הווב של dropbox) - כך תוכלו להמנע מטעויות (כמו זו בהערה מס' 1). במידה ועבר שבוע מזמן שיתוף התיקייה ולא אושר שיתוף ע"י חשבון הבודקים - יש להתריע על כך.

3. לאור 2+1, מומלץ לייצר את תיקיית ההגשות באופן מיידי ולבצע שיתוף שלה עם חשבון הבודקים (לאחר מכן תוכלו להכניס אליה את ההגשות שלכם באופן אסינכרוני). זאת כדי למנוע עוגמת נפש מצדכם ולאפשר לבודקים לתת לכם מענה בזמן לפני דדליין ההגשה של התרגיל הראשון.

4. יש להעתיק את כל תיקיית הפרויקט כמו שהיא לתוך תיקיית ההגשה שלכם ב dropox-מלבד תתי תיקיות ה debug-שיש למחוק.

5. מומלץ לוודא שלא מחקתם בטעות את קבצי ה source-שלכם בתהליך ואת קובץ ה sln.-שנמצא בהיררכיה הראשית של הפרויקט.

6. מומלץ לוודא שהפרויקט שלכם מתקמפל משורת הפקודה, כי כך ייבדק התרגיל שלכם.
איך עושים את זה?

מריצים את msbuild על קובץ ה sln-שנמצא בהיררכיה הראשית של הפרויקט) זהו למעשה exe שמקבל ארגומנט אחד והוא שם ה sln-שלכם .(כתוצאה תיווצר תיקייה חדשה בשם debug בהיררכיה הראשית של עץ התיקיות של הפרויקט ובה קובץ ההרצה של התוכנה שלכם.

איפה מוצאים את msbuild? הוא אמור להגיע עם ההתקנה שלVS . מקום טוב לחפש בו הוא:

C:\Program Files (x86)\MSBuild\14.0\Bin\MsBuild.exe

עד כדי מספר הגרסא וכו.'

7. דרך אחרת לוודא את תקינות התיקייה היא פשוט ללחוץ לחיצה כפולה על קובץ ה sln-**במיקום החדש שלו** לאחר ההעתקה של הפרויקט. במקרה הזה ייפתח visual studio שלכם ותוכלו לקמפל, להריץ ולראות שהכל עובד.

8. בכל מקרה על אחריותכם לבצע לפחות את אחת מהבדיקות כדי להימנע מטעויות מיותרות...

9. יש לפתח ולבדוק את הקוד שלכם בVisual Studio 2010 – זוהי הגרסא המותקנת במעבדות המחשבים באוניברסיטה.

אנא הקפידו על נהלי ההגשה במדויק. תרגילים שלא יעמדו בדרישות ההגשה הטכנית במדויק, יורד להם ציון של 5 נקודות בתרגיל (באופן מצטבר, כלומר שם תיקייה לא נכון + אי הגשת קבצים נכונים = הורדה של 10 נקודות), והם יתבקשו לשלוח את התרגיל פעם נוספת, מתוקן.

<u>שגיאות נפוצות:</u>

1. לא צורף קובץ word שמכיל את הקוד

2. שם התקיה שונה מ-ex#

3. העתקה לא טובה של תקיות הפרויקט

4. עבודה ב visual studio בגרסא שונה מ-2010

<u>הנחיות לגבי סגנון – שימו לב שהנחיות אלו הן בגדר חובה וחלק מהניקוד בבדיקה הלבנה יתייחס אליהן.</u>

<u>מוסכמות כתיבת קוד</u>

I. Comments

Every function in your program should have a small header which contains:

1. Parameters – all function parameters and a short description for each one of them. Also state if this parameter is used for output
2. Returns – describe the return value
3. Description – a short description of the function

Every file in the project (.h/.c) will have a small header which contains:

1. Author – the full name and ID of the author/s
2. Project – the name of the project that this file is belong to (e.g “Exercise 1”)
3. Using – all files (modules) that this file is using (exclude standard library files)
4. Description – a short description of the module implemented by this file.

II. Naming

Use meaningful names for functions, parameters and variables. Don’t be afraid of long names like “InsertItemToQueue()” or “LastActiveItemIndex”. Follow the “Say it with code” principle.

To make your code aesthetic, you are encouraged to use the following style for naming:

1. Function names: Each word starts with a capital letter, and no space between words.
2. variable names: lower case, words separated by underscore.
3. macros & constants: Use upper case and underscore as a separator for (e.g. “MAX_ITEMS_IN_QUEUE”)

File names should be the module name with the *.c / *.h extension, except for the file containing the main function which you may name as you please with meaning, e.g. “main”/”programStart”/”killThoseSubmarinesAlready”.

III. File Structure

File structure for all files should be:

1. File Header
2. Library Includes
3. Project Includes
4. Macros & definitions
5. Declarations
6. Implementation (*.c files only)

Note that not every file will contain all the above sections but it is important to keep this order. Place a separator between the various sections of the file.

<u>ועוד קצת...</u>

Don’t document the obvious! Put extra comments inside functions only if it is not easily understood from the code.

Your project should be decomposed into one or more modules. A module contains part of the entire project code that has some common purpose.
Each module is implemented by a *.h & *.c files.
The header file is the module interface. It contains all the declarations and code that any other module may need in order to use this module.
Any code that is not absolutely necessary for other modules to use this module should be placed in the .c file. The .c file is the module implementation. It contains all the .internal declarations and definitions of the module and the module implementation code
When a module is using another module we say that it is “dependent” on the other module. Avoid circular dependency between modules.

Coding Correctly

- Initiate all pointers to NULL.
- Use dynamically allocated memory to store data which his size is unknown at compilation time. **Unless you were told otherwise, You are not allowed to assume max size for the data.**
- System Resources – When using such a resource never assume that the access to such a resource works.
  Also, don’t forget to **release dynamically allocated memory, file handles etc!**

- I/O validation – Never assume that I/O is valid.

- Document your code. Its hard not to repeat this since in practice software maintenance is generally the most costly portion of the software development life cycle.