פקולטה: מדעי הטבע
מחלקה: מדעי המחשב
שם הקורס: תכנות מערכות א
קוד הקורס: 7029810, כל הקבוצות
מועד __**א'**___ סמסטר: ___**א'** __
תאריך הבחינה: 2.2.2023
משך הבחינה: שעתיים
שם המרצים: ד"ר גיל בן-ארצי, ד"ר אסף חוגי
מתרגלים: יבגני נייטרמן הרשקוביץ, אלמוג שור, חרות סטרמן, חיה לוינגר

- יש לענות על כל השאלות.
- אין להשתמש בטלפונים.
- אסור להשתמש בכל חומר עזר.
- בשאלות הבנת קוד וזיהוי שגיאות, יש להסביר בפירוט מה גורם לשגיאה.
- בשאלות התכנות, יש לכתוב קוד נכון ומסודר לפי כללי התכנות שנלמדו בהרצאות ובתרגולים. יש לכתוב הערות מפורטות בעברית או באנגלית, המסבירות את אופן הפתרון.

בהצלחה!!!

שאלה 1 (14 נק')

א. האם יש טעויות בקוד שלהלן? אם כן יש לתקן אותם.

ב. מה הפונקציה הבאה תדפיס ?

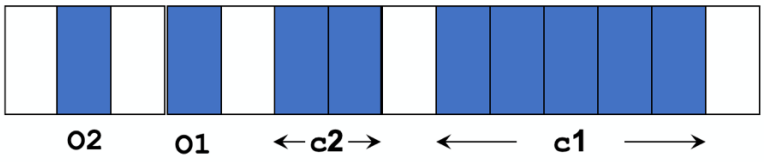
```
#include<stdio.h>

void get_char(){
static int i=0;
i++;
char c='a';
return c+i;
}

void print_char(char c) {
    printf("%c ",c);
}

void print_two_chars(char c1,char c2) {
    printf("%c %c ",c1,c2);
}


int main() {
    char c1 = get_char();
    char c2 = get_char();
    printf("1: \n");
    print_two_chars(c1,c2);
    printf("\n");
    printf("2: \n");
    print two_chars(get_char(),get_char());

    printf("\n");
    printf("3: \n");
    if (get_char()>'d' || get_char()<'q') {
        print_char(get_char());
        print_char(get_char());
    }
    printf("\n");
    printf("4: \n");
    if (get_char()>'d' && get_char()<'q') {
        print_char(get_char());
        print_char(get_char());
    }
    printf("\n");
    return 0;

}
```

שאלה 2 (31 נק')

א. (15 נק') נתונה מפת הביטים הבאה במשתנה בשם status. יש לבצע את הפעולות הבאות על המספרים אשר מופיעים ב c1 c2 לפי הערכים אשר נמצאים ב o1 o2 ולהדפיס את התוצאה. 00 – חיבור, 01 – חיסור, 10 – כפל, 11 – חילוק. יש למשש את הסעיף בעזרת masks ופעולות bitwise

ב. (8 נקודות) יש למשש את סעיף א בעזרת bitfields.

ג. (8 נקודות) יש לכתוב תוכנית שמקבלת מהמשתמש 2 מספרים עשרוניים, ומדפיסה מספר בינרי המהווה את פעולת ה bitwise xor בינהם, ביט אחר ביט.



<u>שאלה 3 (24 נק')</u>

יש לכתוב מבנה נתונים מסוג תור המחזיק מספרים שלמים. התור ייוצג ע"י מבנה queue אותו אתם תגדירו בעצמכם. על התור לתמוך בpush וpop גודל התור אינו מוגבל. במידה ורוצים לעשות pop לתור ריק על הפונקציה להחזיר -1.

א. הגדירו את מבנה הנתונים של התור (8 נק)

```
Typedef struct QUEUE_ {
.....
} queue, *pqueue;
```

ב. ממשו את push ו - pop (8 נק)

```
Void push(queue q, int num);
Int pop(queue q);
```

ג. רשמו פונקציה main קצרה המשתמשת בתור שכתבתם (8 נק)

<u>שאלה 4 (31 נק')</u>

בשאלה זו זה נבנה מערך אדפטיבי **כללי**. מערך אדפטיבי כללי הינו מערך אשר משנה את גודלו כך שכל פנייה לאינדקס אי שלילי (אפס ומעלה) היא תקינה. כאשר פונים לאינדקס בו לא הושם ערך, אז מוחזר NULL.

להלן תיאור אופן הפעולה של חמש המתודות של המערך האדפטיבי

CreateAdptArray	מאתחלת מערך ריק (כלומר ללא איברים)
DeleteAdptArray	משחררת את הזיכרון של האובייקט (כולל איבריו)
SetAdptArrayAt	מקבלת אינדקס ואיבר ושומרות **עותק** של האיבר במקום המבוקש. משחררת את האיבר הישן אם קיים.
GetAdptArrayAt	מקבלת אינדקס ומחזירה **עותק** של האיבר במיקום זה
GetAdptArraySize	מחזירה את גודל המערך (1- כאשר המערך לא אותחל בהצלחה)

בכל מקרה של כישלון יש להחזיר FAIL או NULL, בהתאם למצב.

להלן החתימות של הפונקציות

```
typedef struct AdptArray_* PAdptArray;
typedef enum Result {FAIL = 0, SUCCESS};
typedef void* PElement;

typedef void(*DEL_FUNC)(PElement);
typedef PElement(*COPY_FUNC)(PElement);

PAdptArray CreateAdptArray(COPY_FUNC, DEL_FUNC);
void DeleteAdptArray(PAdptArray);
Result SetAdptArrayAt(PAdptArray, int, PElement);
PElement GetAdptArrayAt(PAdptArray, int);
int GetAdptArraySize(PAdptArray);
```

בחרנו לממש את המבנה ע"י מערך בגודל משתנה. כלומר נשמור מערך של מצביעים כלליים ובכל פעם שנקבל הצבה לאינדקס גדול יותר מהמערך שלנו אז נעתיק את המערך לזיכרון גדול מספיק כך שפנייה זאת תהיה חוקית. נצטרך כמובן לשמור את גודל המערך הנוכחי במבנה ובנוסף מצביעים לפונקציות המשתמש להעתקה ומחיקה של איברים.

בעמודים הבאים נתון לכם מימוש חלקי של המבנה. כאשר תתבקשו לממש אז יש לכתוב מאפס וכאשר תתבקשו להשלים יש להוסיף קוד במקומות החסרים.

א. (11 נק') השלימו את המימוש של struct AdptArray_ וממשו את CreateAdptArray.

ב. (10 נק') השלימו את המימוש של SetAdptArrayAt.

ג. (10 נק') ממשו את הפונקציה DeleteAdptArray.

```
Typedef struct BOOK_{
Char *name;
Int serial_number;
} book, *pbook;
```

```
typedef struct AdptArray_
{
        int ArrSize;
        PElement* pElemArr;
        …
}AdptArray;
```

```
Result SetAdptArrayAt(PAdptArray pArr, int idx, PElement pNewElem)
{
        PElement* newpElemArr;
        if (pArr == NULL)
                return FAIL;
        if (idx >= pArr->ArrSize)
        {
                // Extend Array
                if ((newpElemArr = (PElement*)malloc((idx + 1), sizeof(PElement))) ==
NULL)
                        return FAIL;
                // Init new array and copy old array to new array
(1)             …

                // Free old array and save new array
                free(pArr->pElemArr);
                pArr->pElemArr = newpElemArr;
        }

        // Delete Previous Elem and Set New Elem
(2)     …

        // Update Array Size
        pArr->ArrSize = (idx >= pArr->ArrSize) ? (idx + 1) : pArr->ArrSize;
        return SUCCESS;
}
```