פקולטה: מדעי הטבע

מחלקה: מדעי המחשב

שם הקורס: מבוא למחשבים ושפת C

קוד הקורס: 2-7028510

תאריך בחינה: **פיתרון בחינה**

משך הבחינה: שעתיים

שם המרצה: ד"ר אופיר פלא

<u>שאלה 1 (15 נק')</u>

א. 11 וזאת מכיוון שהמימד האחרון של המערך מוטל ישירות לפוינטר. לכן הטיפוס האמיתי שנשלח הוא:

```
int (*arr)[COLS]
```

כלומר, פוינטר למערך של int באורך COLS. לכן התוספת של -2 תתבצע ביחידות הללו ונגיע למספר 11.

ב. נשנה את המימוש ל:

```
return (*((int*)arr+n));
```

כלומר, נטיל את arr לפני התוספת של ה-n.

שאלה 2 (15 נק')

א. השורה:

```
STR_SWAP (str1,str2);
```

תוחלף ע"י הpre-processor ל:

```
{char* t=str1; str1=str2; str2=t;};
```

מכיוון ש-str1 וstr2- הם מערכים ולא פוינטרים, אי אפשר להכניס לתוכם ערך אחר.

ב. הפלט יהיה:

```
B A
B A
```

מכיוון שהפונקציה מחליפה בין הפוינטרים הזמניים שהועברו אליה בהעתקה ולכן לא יהיה חילוף בין

המערכים ב stack של ה main.

שאלה 3 (20 נק')

כתובת	מיקום
a._pb (14)	לא מוגדר
&pb (15)	מחסנית
pb (15)	ערימה דינמית
&(pb->_arr[3]) (15)	לא מוגדר
&(pb->_arr[3]) (18)	מחסנית
arr+3 (18)	מחסנית
&pa (23)	מחסנית
pa->_pb (23)	ערימה דינמית

ב.

```
free(pb);
free(pa);
```

שאלה 4 (50 נק')

א.

```
size_t rand_ind(size_t minVal, size_t maxVal) {
```

```
     return minVal + (rand() % (maxVal+1));
}

int is_sorted(const void *base, size_t nmemb, size_t size,
              int (*compar)(const void *, const void *)) {
     size_t i;
     for (i=0; i<nmemb-1; ++i, base+= size) {
         if (compar(base,base+size)>0) return 0;
     }
     return 1;
}

void swap_slow_sort(void *base, size_t nmemb, size_t size,
                    int (*compar)(const void *, const void *)) {
     while (!is_sorted(base, nmemb, size, compar)) {
         memswap_ptr( base+size*rand_ind(0,nmemb-1),
                              base+size*rand_ind(0,nmemb-1),
                              size);
     }
}
```

ב. הערה: יתקבלו גם תשובות קצרות יותר אשר לפחות מכילים בדיקה של כל פונקציות העזר מסעיף א (חוץ מהמספרים הרנדומיים) ומכילים את INT_MAX,INT_MIN

```
int int_compare(const void* a, const void* b) {
     const int* ia= (const int*)a;
     const int* ib= (const int*)b;
     if ((*ia)>(*ib)) return 1;
     if ((*ia)<(*ib)) return -1;
     return 0;
}



TEST(is_sorted, simpleTests) {

     {
         int arr[]= {1};
         EXPECT_TRUE(is_sorted(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare));
     }

     {
         int arr[]= {1,2};
         EXPECT_TRUE(is_sorted(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare));
     }

     {
         int arr[]= {2,-1};
         EXPECT_FALSE(is_sorted(arr, sizeof(arr)/sizeof(*arr),
sizeof(*arr), &int_compare));
     }

}
```

```
TEST(is_sorted, limitsTests) {

     {
         int arr[]= {INT_MIN};
         EXPECT_TRUE(is_sorted(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare));
     }

     {
         int arr[]= {INT_MIN,INT_MAX};
         EXPECT_TRUE(is_sorted(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare));
     }

     {
         int arr[]= {3,INT_MAX};
         EXPECT_TRUE(is_sorted(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare));
     }

     {
         int arr[]= {INT_MAX,INT_MIN};
         EXPECT_FALSE(is_sorted(arr, sizeof(arr)/sizeof(*arr),
sizeof(*arr), &int_compare));
     }

     {
         int arr[]= {INT_MAX,3};
         EXPECT_FALSE(is_sorted(arr, sizeof(arr)/sizeof(*arr),
sizeof(*arr), &int_compare));
     }

}



TEST(swap_slow_sort, simpleTests) {

     {
         int arr[]= {1};
         int exp_arr[]= {1};
         swap_slow_sort(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare);
         for (size_t i=0; i<sizeof(arr)/sizeof(*arr); ++i) {
               EXPECT_EQ(exp_arr[i], arr[i]);
         }
     }

     {
         int arr[]= {1,2};
         int exp_arr[]= {1,2};
         swap_slow_sort(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare);
         for (size_t i=0; i<sizeof(arr)/sizeof(*arr); ++i) {
               EXPECT_EQ(exp_arr[i], arr[i]);
         }
```

```
    }

    {
        int arr[]=     {-2,1};
        int exp_arr[]= {-2,1};
        swap_slow_sort(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare);
        for (size_t i=0; i<sizeof(arr)/sizeof(*arr); ++i) {
              EXPECT_EQ(exp_arr[i], arr[i]);
        }
    }

    {
        int arr[]=     { 2, -1, 10, -8 };
        int exp_arr[]= { -8, -1, 2, 10 };
        swap_slow_sort(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare);
        for (size_t i=0; i<sizeof(arr)/sizeof(*arr); ++i) {
              EXPECT_EQ(exp_arr[i], arr[i]);
        }
    }

    {
        int arr[]=     { 0, 4, 3, 2, 5, 7, 1, 6 };
        int exp_arr[]= { 0, 1, 2, 3, 4, 5, 6, 7 };
        swap_slow_sort(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare);
        for (size_t i=0; i<sizeof(exp_arr)/sizeof(*exp_arr); ++i) {
              EXPECT_EQ(exp_arr[i], arr[i]);
        }
    }

    {
        int arr[]=     { -2, 1, -3 };
        int exp_arr[]= { -3, -2, 1 };
        swap_slow_sort(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare);
        for (size_t i=0; i<sizeof(exp_arr)/sizeof(*exp_arr); ++i) {
              EXPECT_EQ(exp_arr[i], arr[i]);
        }
    }

    {
        int arr[]=     { 2, -1, 10, -8, 500 };
        int exp_arr[]= { -8, -1, 2, 10, 500 };
        swap_slow_sort(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare);
        for (size_t i=0; i<sizeof(exp_arr)/sizeof(*exp_arr); ++i) {
              EXPECT_EQ(exp_arr[i], arr[i]);
        }
    }

    {
        int arr[]=     { 0, 3, 2, 5, 7, 1, 6 };
        int exp_arr[]= { 0, 1, 2, 3, 5, 6, 7 };
        swap_slow_sort(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare);
```

```
        for (size_t i=0; i<sizeof(exp_arr)/sizeof(*exp_arr); ++i) {
              EXPECT_EQ(exp_arr[i], arr[i]);
        }
    }
}



TEST(swap_slow_sort, limitsTests) {

    {
        int arr[]=     { 0, 3, INT_MIN, 2, 5, INT_MAX, 7, 1, 6 };
        int exp_arr[]= { INT_MIN, 0, 1, 2, 3, 5, 6, 7, INT_MAX };
        swap_slow_sort(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare);
        for (size_t i=0; i<sizeof(exp_arr)/sizeof(*exp_arr); ++i) {
              EXPECT_EQ(exp_arr[i], arr[i]);
        }
    }

    {
        int arr[]=     { INT_MIN,INT_MAX};
        int exp_arr[]= { INT_MIN,INT_MAX };
        swap_slow_sort(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare);
        for (size_t i=0; i<sizeof(exp_arr)/sizeof(*exp_arr); ++i) {
              EXPECT_EQ(exp_arr[i], arr[i]);
        }
    }

    {
        int arr[]=     { 0, 3, INT_MIN, 2, 5, INT_MAX, 7, 1, 6 };
        int exp_arr[]= { INT_MIN, 0, 1, 2, 3, 5, 6, 7, INT_MAX };
        swap_slow_sort(arr, sizeof(arr)/sizeof(*arr), sizeof(*arr),
&int_compare);
        for (size_t i=0; i<sizeof(exp_arr)/sizeof(*exp_arr); ++i) {
              EXPECT_EQ(exp_arr[i], arr[i]);
        }
    }
}
```

בהצלחה!