ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

Τμήμα Πληροφορικής & Τηλεπικοινωνιών

Ασκήσεις-Παιχνίδια για Επίλυση (Προγραμματισμός Ι) – bonus

Παράδοση: Upload μέσω του e-class (οι ασκήσεις δεν είναι υποχρεωτικές).

Καταληκτική Ημερομηνία/Ώρα Αποστολής: Κυριακή, 14/01/2024 (23:59:59).

Οδηγίες Υποβολής: Το αρχείο που θα περιέχει τον κώδικα με τη λύση του κάθε παιχνιδιού θα πρέπει να ακολουθεί την παρακάτω ονοματολογία: **`Epitheto_Onoma_game1.c`** , **`Epitheto_Onoma_game2.c`** και **`Epitheto_Onoma_game3.c`** όπου `Epitheto` είναι το επίθετο του κάθε φοιτητή με λατινικούς χαρακτήρες και `Onoma` είναι το όνομά του, αντίστοιχα.

Π.χ. για τον υποτιθέμενο φοιτητή Νίκο Κούλη, ο κώδικας των ασκήσεων-εφαρμογών θα πρέπει να βρίσκεται στα αρχεία με όνομα: **`Koulis_Nikos_game1.c`** , **`Koulis_Nikos_game2.c`** και **`Koulis_Nikos_game3.c`**

Τα τρία παραπάνω αρχεία (ή τα δύο ή το ένα, ανάλογα με πόσα θα ασχοληθείτε) θα πρέπει να περιέχονται σε συμπιεσμένο αρχείο (`.rar` ή `.zip`) βάσει της ονοματολογίας: `Epitheto_Onoma_games.rar` ή `Epitheto_Onoma_games.zip`.

Π.χ. για τον προαναφερθέντα φοιτητή, το αρχείο αυτό θα είναι το: `Koulis_Nikos_games.rar` ή `Koulis_Nikos_games.zip` .

Ασκήσεις που θα παραδοθούν, αλλά **δεν θα ακολουθούν τον παραπάνω κανόνα ονοματολογίας, θα αγνοηθούν**.

Παιχνίδι 1:

Ας υλοποιήσουμε ένα παιχνίδι μνήμης με χρήση τράπουλας. Το παιχνίδι θα παίζεται με 2 ή περισσότερους παίκτες (max 5 παίχτες) και μία τράπουλα. Για όσους δεν γνωρίζουν, η τράπουλα αποτελείται από 52 φύλλα, που χωρίζονται σε 4 "σειρές" (μπαστούνια ♠, καρό ♦, σπαθιά ♣ και κούπες ♥), ενώ κάθε "σειρά" περιλαμβάνει τους αριθμούς από το 2 έως και το 10, τις φιγούρες: Βαλές (`J`), Ντάμα (`Q`), Ρήγας (`K`) και τον άσσο (`A`).

Γενικοί κανόνες του παιχνιδιού:

Με την έναρξη του παιχνιδιού, ζητείται ο συνολικός αριθμός των παικτών. Έπειτα από σχετική απάντηση (αποδεκτές τιμές: 2 έως 5), θεωρούμε ότι οι δηλωθέντες παίκτες θα παίζουν στο εξής ο ένας μετά τον άλλον, βάσει ηλικίας, ξεκινώντας από τον μεγαλύτερο προς τον μικρότερο. Στη συνέχεια, η τράπουλα «ανακατεύεται» και τα 52 φύλλα τοποθετούνται «κλειστά» στο τραπέζι (δηλαδή έχουν κρυμμένη την όψη που αναγράφει το κάθε τραπουλόχαρτο, ώστε οι παίχτες να μην γνωρίζουν ποιο είναι το κάθε φύλλο πάνω στο τραπέζι). Για λόγους ευκολίας στη διάδραση παίχτη-ηλεκτρονικού παιχνιδιού, θεωρούμε ότι τα φύλλα τοποθετούνται κλειστά σε διάταξη διδιάστατου πίνακα `4x13` (4 γραμμές και 13 στήλες).

Όταν έρθει η σειρά του κάθε παίκτη, αυτός καλείται να ανοίξει ένα ζευγάρι κλειστών φύλλων. Αν τα φύλλα που θα ανοίξουν έχουν το ίδιο σύμβολο (2, ..., 10, J, Q, K ή A), τότε ο παίκτης κερδίζει τους πόντους που αντιστοιχούν στην αξία των φύλλων (βλέπε ιδιαίτερους κανόνες στο Παράρτημα), τα φύλλα αυτά μένουν ανοιχτά στο τραπέζι και παίζει ο επόμενος παίκτης (πλην ελαχίστων εξαιρέσεων, βάσει των ιδιαίτερων κανόνων του Παραρτήματος). Αν τα επιλεγμένα φύλλα δεν έχουν το ίδιο σύμβολο, τότε τοποθετούνται πάλι κλειστά στην ίδια θέση και το παιχνίδι συνεχίζεται με τον επόμενο παίκτη. Το παιχνίδι ολοκληρώνεται όταν ανοιχτούν όλα τα φύλλα,

οπότε ανακηρύσσεται νικητής ο παίκτης που έχει συγκεντρώσει τους περισσότερους πόντους (σε περίπτωση πολλαπλής ισοβαθμίας έχουμε και πολλαπλούς νικητές).

Το παιχνίδι θα πρέπει να κάνει έλεγχο στα δεδομένα που εισάγει ο χρήστης και θα ζητά επανάληψη της εισόδου σε περίπτωση μη αποδεκτών επιλογών όπως:

- Λάθος δεδομένα εκκίνησης του παιχνιδιού (π.χ. μη αποδεκτός αριθμός παικτών)
- Λάθος στοιχεία για επιλογή φύλλου (π.χ. άνοιξε το φύλλο που βρίσκεται στην $5^η$ γραμμή και στην $20^η$ στήλη, που είναι – προφανώς – εκτός του διδιάστατου πίνακα `4x13`)
- Επιλογή για άνοιγμα φύλλου που είναι ήδη ανοικτό

Σημείωση: Για την αποδοτικότερη υλοποίηση του παιχνιδιού, ο κώδικας θα πρέπει να «σπάσει» σε συναρτήσεις. Στο τέλος, παρουσιάζεται μία ενδεικτική προσέγγιση του ορισμού απαραίτητων συναρτήσεων και όποιος θέλει μπορεί να ακολουθήσει τον συγκεκριμένο «σκελετό», ενώ για τη διευκόλυνσή σας κατά τη διάρκεια της υλοποίησης, προτείνεται ο διδιάστατος πίνακας με τα φύλλα της τράπουλας να τυπώνεται κάθε φορά 2 φορές (μία με κλειστά φύλλα, όπως ορίζουν οι κανόνες του παιχνιδιού, αλλά και μία με όλα τα φύλλα ανοιχτά, για να μπορέσετε να εξαντλήσετε όλα τα ενδεχόμενα κατά το testing, χωρίς να χρειάζεται κάθε φορά να «μαντεύετε»).

ΠΑΡΑΡΤΗΜΑ ΠΑΙΧΝΙΔΙΟΥ 1

Ειδικά φύλλα:

- Αν κάποιος παίκτης ανοίξει δύο Βαλέδες (J), τότε κερδίζει τους αντίστοιχους πόντους και ο επόμενος παίκτης χάνει τη σειρά του.
- Αν κάποιος παίκτης ανοίξει δύο Ντάμες (Q), κερδίζει τους αντίστοιχους πόντους και παίζει ξανά.
- Αν κάποιος παίκτης ανοίξει έναν Ρήγα (K) ή/και έναν Άσσο (A), τότε μπορεί να ανοίξει και ένα τρίτο φύλλο. Αν δύο από τα τρία φύλλα ταιριάζουν μεταξύ τους, τότε αυτά μένουν ανοικτά και ο παίκτης κερδίζει τους αντίστοιχους πόντους. Τα φύλλα που δεν ταιριάζουν κλείνουν. Σε περίπτωση που το τρίτο φύλλο που θα ανοίξει είναι Ρήγας (K) ή Άσσος (A) αλλά δεν δημιουργείται ζευγάρι, δεν επαναλαμβάνεται ο ίδιος κανόνας (δηλ. δεν ανοίγει τέταρτο φύλλο).

Αξίες φύλλων:

- Ζευγάρι Άσσων (A) έχει αξία 20
- Ζευγάρι από φιγούρες, δηλαδή Βαλέδες (J), Ντάμες (Q) ή Ρηγάδες (K) έχει αξία 15
- Ζευγάρι από τα υπόλοιπα φύλλα, έχει αξία ίση με τον αριθμό του ενός φύλλου (π.χ. ζευγάρι από 8άρια, έχει αξία 8 ενώ ζευγάρι από 2άρια έχει αξία 2).

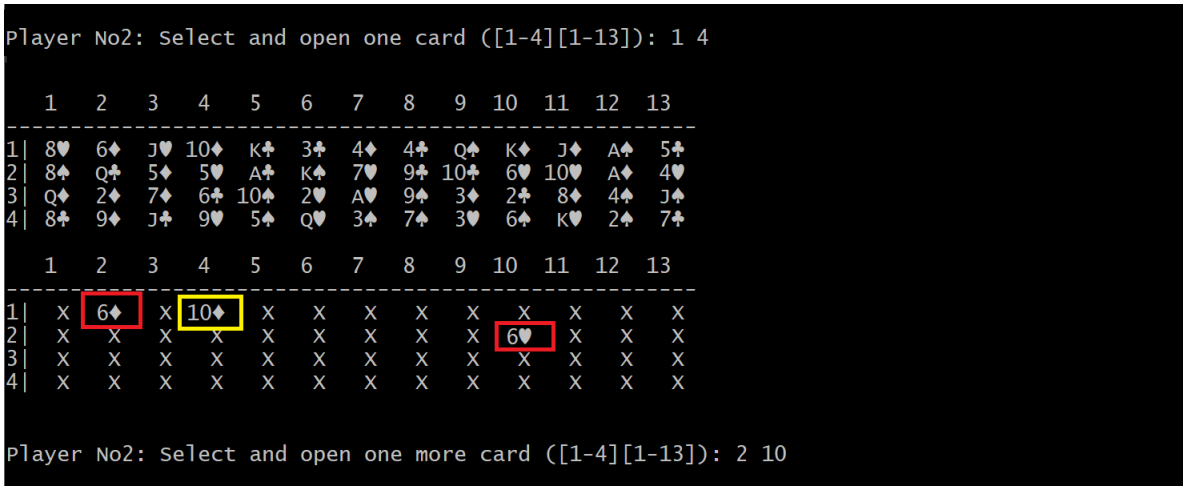
Ενδεικτικό «τρέξιμο»:

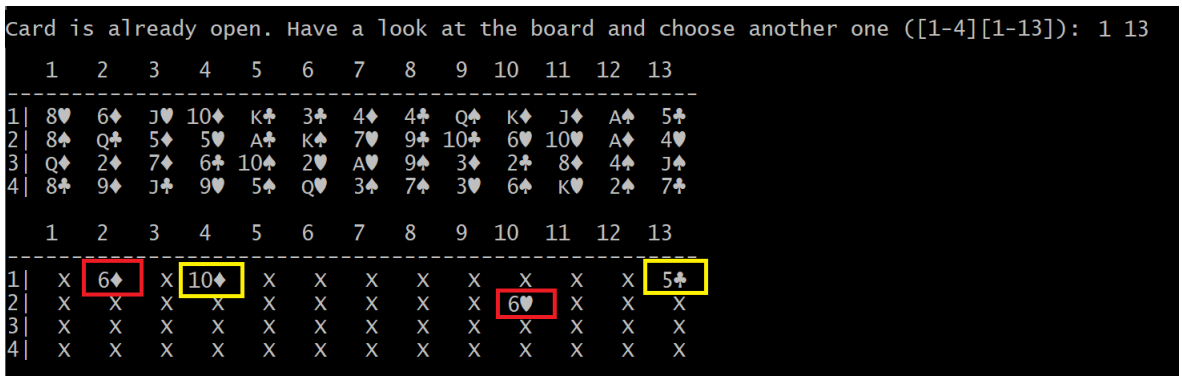
```
nitse@DESKTOP-6P8P96L ~
$ gcc -o cards cards.c

nitse@DESKTOP-6P8P96L ~
$ ./cards

*********************************************************************************
****************** W E L C O M E   T O   L A S   V E G A S ******************
*********************************************************************************
------------------------------ PRO-1 CARDS' GAME --------------------------------

Give me the number of players [2-5]: 2

   1   2   3   4   5   6   7   8   9  10  11  12  13
-----------------------------------------------------
1| 8♥  6♦  J♥ 10♦  K♣  3♣  4♦  4♣  Q♠  K♦  J♦  A♠  5♣
2| 8♠  Q♣  5♦  5♥  A♣  K♠  7♥  9♣ 10♣  6♥ 10♥  A♦  4♥
3| Q♦  2♦  7♦  6♣ 10♠  2♥  A♥  9♠  3♦  2♣  8♦  4♠  J♠
4| 8♣  9♦  J♣  9♥  5♠  Q♥  3♠  7♠  3♥  6♠  K♥  2♠  7♣

   1   2   3   4   5   6   7   8   9  10  11  12  13
-----------------------------------------------------
1|  X   X   X   X   X   X   X   X   X   X   X   X   X
2|  X   X   X   X   X   X   X   X   X   X   X   X   X
3|  X   X   X   X   X   X   X   X   X   X   X   X   X
4|  X   X   X   X   X   X   X   X   X   X   X   X   X

Player No1: Select and open a card ([1-4][1-13]): 1 2
```

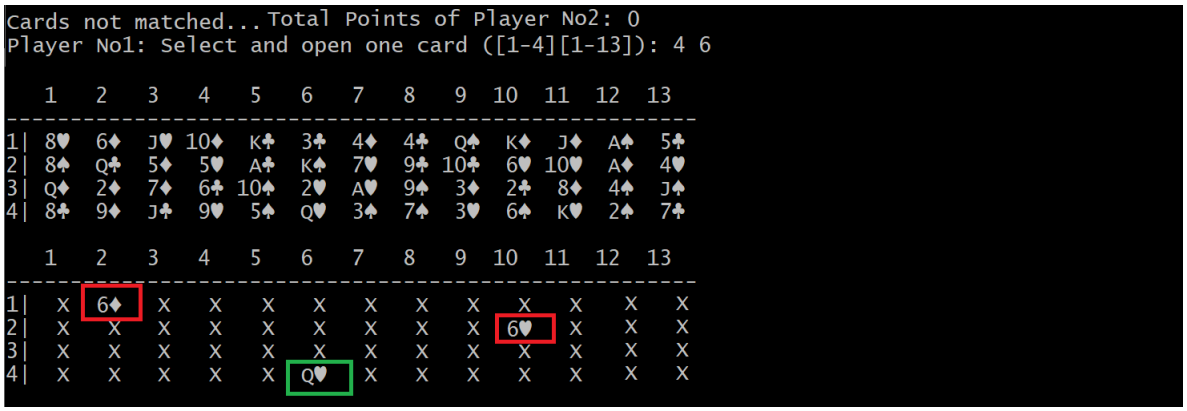
Το board με όλα τα φύλλα ανοιχτά, τυπώνεται μόνο για λόγους testing. Όταν ολοκληρωθεί η υλοποίηση θα πρέπει να αφαιρεθεί...

```
   1   2   3   4   5   6   7   8   9  10  11  12  13
-----------------------------------------------------
1| 8♥  6♦  J♥ 10♦  K♣  3♣  4♦  4♣  Q♠  K♦  J♦  A♠  5♣
2| 8♠  Q♣  5♦  5♥  A♣  K♠  7♥  9♣ 10♣  6♥ 10♥  A♦  4♥
3| Q♦  2♦  7♦  6♣ 10♠  2♥  A♥  9♠  3♦  2♣  8♦  4♠  J♠
4| 8♣  9♦  J♣  9♥  5♠  Q♥  3♠  7♠  3♥  6♠  K♥  2♠  7♣

   1   2   3   4   5   6   7   8   9  10  11  12  13
-----------------------------------------------------
1|  X  6♦   X   X   X   X   X   X   X   X   X   X   X
2|  X   X   X   X   X   X   X   X   X   X   X   X   X
3|  X   X   X   X   X   X   X   X   X   X   X   X   X
4|  X   X   X   X   X   X   X   X   X   X   X   X   X
```

```
Player No1: Select and open one more card ([1-4][1-13]): 2 10

   1   2   3   4   5   6   7   8   9  10  11  12  13
-----------------------------------------------------
1| 8♥  6♦  J♥ 10♦  K♣  3♣  4♦  4♣  Q♠  K♦  J♦  A♠  5♣
2| 8♠  Q♣  5♦  5♥  A♣  K♠  7♥  9♣ 10♣  6♥ 10♥  A♦  4♥
3| Q♦  2♦  7♦  6♣ 10♠  2♥  A♥  9♠  3♦  2♣  8♦  4♠  J♠
4| 8♣  9♦  J♣  9♥  5♠  Q♥  3♠  7♠  3♥  6♠  K♥  2♠  7♣

   1   2   3   4   5   6   7   8   9  10  11  12  13
-----------------------------------------------------
1|  X  6♦   X   X   X   X   X   X   X   X   X   X   X
2|  X   X   X   X   X   X   X   X   X  6♥   X   X   X
3|  X   X   X   X   X   X   X   X   X   X   X   X   X
4|  X   X   X   X   X   X   X   X   X   X   X   X   X

Cards matched!!! +6 Points!!! Total Points of Player No1: 6
```


Player No2: Select and open one card ([1-4][1-13]): 1 4
1 2 3 4 5 6 7 8 9 10 11 12 13
1| 8♥ 6♦ J♥ 10♦ K♣ 3♣ 4♦ 4♣ Q♠ K♦ J♦ A♠ 5♣
2| 8♠ Q♣ 5♦ 5♥ A♣ K♠ 7♥ 9♣ 10♣ 6♥ 10♥ A♦ 4♥
3| Q♦ 2♦ 7♦ 6♣ 10♠ 2♥ A♥ 9♠ 3♦ 2♣ 8♦ 4♠ J♠
4| 8♣ 9♦ J♣ 9♥ 5♠ Q♥ 3♠ 7♠ 3♥ 6♠ K♥ 2♠ 7♣
1 2 3 4 5 6 7 8 9 10 11 12 13
1| X 6♦ X 10♦ X X X X X X X X X
2| X X X X X X X X X 6♥ X X X
3| X X X X X X X X X X X X X
4| X X X X X X X X X X X X X
Player No2: Select and open one more card ([1-4][1-13]): 2 10


Card is already open. Have a look at the board and choose another one ([1-4][1-13]): 1 13
1 2 3 4 5 6 7 8 9 10 11 12 13
1| 8♥ 6♦ J♥ 10♦ K♣ 3♣ 4♦ 4♣ Q♠ K♦ J♦ A♠ 5♣
2| 8♠ Q♣ 5♦ 5♥ A♣ K♠ 7♥ 9♣ 10♣ 6♥ 10♥ A♦ 4♥
3| Q♦ 2♦ 7♦ 6♣ 10♠ 2♥ A♥ 9♠ 3♦ 2♣ 8♦ 4♠ J♠
4| 8♣ 9♦ J♣ 9♥ 5♠ Q♥ 3♠ 7♠ 3♥ 6♠ K♥ 2♠ 7♣
1 2 3 4 5 6 7 8 9 10 11 12 13
1| X 6♦ X 10♦ X X X X X X X X 5♣
2| X X X X X X X X X 6♥ X X X
3| X X X X X X X X X X X X X
4| X X X X X X X X X X X X X


Cards not matched... Total Points of Player No2: 0
Player No1: Select and open one card ([1-4][1-13]): 4 6
1 2 3 4 5 6 7 8 9 10 11 12 13
1| 8♥ 6♦ J♥ 10♦ K♣ 3♣ 4♦ 4♣ Q♠ K♦ J♦ A♠ 5♣
2| 8♠ Q♣ 5♦ 5♥ A♣ K♠ 7♥ 9♣ 10♣ 6♥ 10♥ A♦ 4♥
3| Q♦ 2♦ 7♦ 6♣ 10♠ 2♥ A♥ 9♠ 3♦ 2♣ 8♦ 4♠ J♠
4| 8♣ 9♦ J♣ 9♥ 5♠ Q♥ 3♠ 7♠ 3♥ 6♠ K♥ 2♠ 7♣
1 2 3 4 5 6 7 8 9 10 11 12 13
1| X 6♦ X X X X X X X X X X X
2| X X X X X X X X X 6♥ X X X
3| X X X X X X X X X X X X X
4| X X X X X Q♥ X X X X X X X

... κτλ κτλ ...

Ενδεικτικός «σκελετός» προγράμματος:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define ROWS 4
#define COLS 13
#define MAX_PLAYERS 5
enum card { SA = 1, S2, S3, S4, S5, S6, S7, S8, S9, S10, SJ, SQ, SK, DA, D2, D3, D4,
D5, D6, D7, D8, D9, D10, DJ, DQ, DK, CA, C2, C3, C4, C5, C6, C7, C8, C9, C10, CJ,
CQ, CK, HA, H2, H3, H4, H5, H6, H7, H8, H9, H10, HJ, HQ, HK };

/* ***** Functions' prototypes ***** */

const char* get_card_name(enum card c);
/* Function to display spades/diamonds/clubs/hearts properly. */

void card_suffling(int table[][COLS]);
/* Shuffling cards function. */

void print_table(int table[][COLS], int table_opened_cards[][COLS]);
/* Function to display the table board (showing open cards, X otherwise). */

void add_points(int points[], int player_id, int win_points);
/* Function to add points in the points[] array (adds "win_points" to the proper
player, defined by "player_id"). */

void print_winners(int points[]);
/* Function to display the winner(s) at the end of the game, based on the points[]
array. */

int select_card(int table[][COLS], int* ptr, int player_id, int no_of_card);
/* Function to select a card to open, returns the selected "row" of the table
(returned value) and (indirectly) the selected col of the table (by using the "ptr"
pointer).
The "no_of_card" can be either 1, 2 or 3, and depends on which card is opening
(first, second or third, respectively) by the player defined by "player_id". */

/* ***** End of functions' prototypes ***** */


int main(void)
{

	int table[ROWS][COLS] = { 0 }; /*All the 52 cards. */
	int table_opened_cards[ROWS][COLS] = { 0 }; /*Set to 1 if the card should be
	open until the end of the game. */
	int points[MAX_PLAYERS] = { 0 }; /*Array to store the points of each player.
	Player's 1 points -> points[0], Player's 2 points -> points[1], etc.*/

	srand(time(NULL));

	card_suffling(table);

	print_table(table, table_opened_cards);

	return 0;
}

void card_suffling(int table[][COLS])
{
	/* Function's code goes here */
}
```

```c
void print_table(int table[][COLS], int table_opened_cards[][COLS])
{
/* ********* REMOVE THE CODE BELOW. It is used to display the cards, for testing
purpose... ********* */
	int i, j;
	printf("\n");
	for (j = 0; j < COLS; j++)
		printf("%4d", j + 1);
	printf("\n-----------------------------------------------------");
	printf("\n1|");
	for (i = 0; i < ROWS; i++)
	{
		for (j = 0; j < COLS; j++)
			printf("%5s ", get_card_name(table[i][j]));
		if (i < ROWS - 1)
			printf("\n%d|", i + 2);
	}
	printf("\n");

/* ********* REMOVE THE ABOVE CODE. It is used to display the cards, for testing
purpose... ********* */

	/* Function's code goes here */
}


void add_points(int points[], int player_id, int win_points)
{
	/* Function to add points in the points[] array (adds "win_points" to the
proper player, defined by "player_id"). */

	/* Function's code goes here */
}


void print_winners(int points[])
{
	/* Function to display the winner(s) at the end of the game, based on the
points[] array. */

	/* Function's code goes here */
}


int select_card(int table[][COLS], int* ptr, int player_id, int no_of_card)
{
	/* Function to select a card to open, returns the selected "row" of the table
(returned value) and (indirectly) the selected "col" of the table (by using the
"ptr" pointer).
The "no_of_card" can be either 1, 2 or 3, and depends on which card is opening
(first, second or third, respectively) by the player defined by "player_id". */

	/* Function's code goes here */

}

const char* get_card_name(enum card c)
{

	switch (c)
	{
	case SA: return "A\u2660";
	case S2: return "2\u2660";
	case S3: return "3\u2660";
	case S4: return "4\u2660";
```

```
            case S5: return "5\u2660";
            case S6: return "6\u2660";
            case S7: return "7\u2660";
            case S8: return "8\u2660";
            case S9: return "9\u2660";
            case S10: return "10\u2660";
            case SJ: return "J\u2660";
            case SQ: return "Q\u2660";
            case SK: return "K\u2660";
            case DA: return "A\u2666";
            case D2: return "2\u2666";
            case D3: return "3\u2666";
            case D4: return "4\u2666";
            case D5: return "5\u2666";
            case D6: return "6\u2666";
            case D7: return "7\u2666";
            case D8: return "8\u2666";
            case D9: return "9\u2666";
            case D10: return "10\u2666";
            case DJ: return "J\u2666";
            case DQ: return "Q\u2666";
            case DK: return "K\u2666";
            case CA: return "A\u2663";
            case C2: return "2\u2663";
            case C3: return "3\u2663";
            case C4: return "4\u2663";
            case C5: return "5\u2663";
            case C6: return "6\u2663";
            case C7: return "7\u2663";
            case C8: return "8\u2663";
            case C9: return "9\u2663";
            case C10: return "10\u2663";
            case CJ: return "J\u2663";
            case CQ: return "Q\u2663";
            case CK: return "K\u2663";
            case HA: return "A\u2665";
            case H2: return "2\u2665";
            case H3: return "3\u2665";
            case H4: return "4\u2665";
            case H5: return "5\u2665";
            case H6: return "6\u2665";
            case H7: return "7\u2665";
            case H8: return "8\u2665";
            case H9: return "9\u2665";
            case H10: return "10\u2665";
            case HJ: return "J\u2665";
            case HQ: return "Q\u2665";
            case HK: return "K\u2665";
            }
}
```

Παιχνίδι 2:

Να υλοποιηθεί το παιχνίδι τράπουλας «black-jack» ακολουθώντας τους παρακάτω κανόνες:

α) Αρχικά, θα πρέπει να γίνεται τράβηγμα ενός φύλλου για τη «μάνα» και δύο φύλλων για τον παίκτη. Αν τα δύο φύλλα του παίκτη είναι «Άσσος» και οποιαδήποτε φιγούρα ή 10, ο παίκτης λέμε ότι κάνει «black jack», οπότε κερδίζει και το πρόγραμμα τερματίζει. Αν είναι ίδια (π.χ. δύο τεσσάρια), το πρόγραμμα να ρωτάει τον παίκτη αν θέλει να τα διαχωρίσει (split) σε δύο ξεχωριστά παιξίματα («χέρια»). Αν απαντήσει ναι, τότε το κάθε φύλλο παίζεται ξεχωριστά, σαν να έπαιζαν δύο διαφορετικοί παίκτες το καθένα. Για κάθε παίξιμο φύλλου, το πρόγραμμα να ρωτάει τον παίκτη αν επιθυμεί να τραβήξει κι άλλο φύλλο μέχρι αυτός να του πει ότι θέλει να σταματήσει. Αν το άθροισμα των φύλλων που έχει τραβήξει ο παίκτης υπερβεί το 21, τότε ο παίχτης χάνει («καίγεται»).
β) Ο παίκτης επιτρέπεται να σταματήσει ή να τραβήξει φύλλο χωρίς κάποιο περιορισμό. Η μάνα είναι υποχρεωμένη να τραβάει φύλλα μέχρι το συνολικό άθροισμα των φύλλων της να υπερβεί το 16, ακόμα και αν σε κάποια φάση έχει μεγαλύτερο άθροισμα από τον παίκτη. Αν υπερβεί το 16 (και όχι το 21), η μάνα είναι υποχρεωμένη να σταματήσει, ακόμα και αν ο παίκτης έχει μεγαλύτερο άθροισμα. Όπως και με τον παίκτη, αν η μάνα υπερβεί το 21 χάνει («καίγεται»).
γ) Αν δεν «καεί» κανένας από τους δύο, κερδίζει αυτός που έχει το μεγαλύτερο συνολικό άθροισμα φύλλων. Αν έχουν το ίδιο άθροισμα, τότε το παιχνίδι λήγει ισόπαλο.

Για το τράβηγμα του φύλλου, να χρησιμοποιήσετε τη συνάρτηση `rand()` η οποία να επιστρέφει την τιμή του φύλλου (πιθανές τιμές φύλλου: θετικός ακέραιος που ανήκει στο [1, 13] με την εξής αντιστοίχιση: «Άσσος»: 1, «Δύο»: 2, …, «Δέκα»: 10, «Βαλές»: 11, «Ντάμα»: 12, «Ρήγας»: 13). Οι «φιγούρες» (Βαλές, Ντάμα και Ρήγας) μετράνε για 10. Ο «Άσσος» μετράει είτε για 1 είτε για 11, ενώ αν ανάμεσα στα φύλλα του παίχτη ή της μάνας υπάρχουν δύο «Άσσοι», το άθροισμά τους είναι αναγκαστικά 12 (μετράει 11 ο ένας και 1 ο άλλος). Το πρόγραμμα να εμφανίζει την τιμή των φύλλων. Θεωρήστε επίσης ότι το παιχνίδι παίζεται με πολλές τράπουλες, οπότε, κατά τη διάρκεια του παιχνιδιού, το κάθε φύλλο (π.χ. ο «άσσος») δεν μπορεί να εμφανιστεί μόνο έως 4 φορές, αλλά και περισσότερες.

Για την καλή οργάνωση του προγράμματος να χρησιμοποιήσετε συναρτήσεις. Για παράδειγμα, να υλοποιήσετε:

α) Μία συνάρτηση που να εμφανίζει το φύλλο που τράβηξε ο παίκτης ή η μάνα.
β) Μία συνάρτηση που να προσομοιώνει το παίξιμο του παίκτη. Για παράδειγμα, να τον ρωτάει αν θέλει να τραβήξει κι άλλο φύλλο ή να σταματήσει. Σε κάθε παίξιμο να εμφανίζεται μήνυμα στην οθόνη για το ποιο είναι το φύλλο που τραβήχτηκε, καθώς και ποιο είναι το νέο συνολικό άθροισμα. Προσοχή: στην περίπτωση που υπάρχει «άσσος» και μπορεί να αποτιμηθεί είτε ως 1 είτε ως 11, θα εμφανίζονται και τα δύο δυνατά αθροίσματα. Αν ο παίχτης σταματήσει σε σημείο που υπάρχουν δύο δυνατά αθροίσματα, προφανώς θα θεωρούμε ως άθροισμα το μεγαλύτερο από τα δύο. Π.χ. αν ο παίχτης έχει ως φύλλα: «δύο», «τέσσερα» και «άσσο», δηλαδή άθροισμα 7 ή 17, και σταματήσει, θεωρούμε ότι το άθροισμα των φύλλων του είναι 17.
γ) Μία συνάρτηση που να προσομοιώνει το παίξιμο της μάνας. Σε κάθε παίξιμο να εμφανίζεται μήνυμα στην οθόνη για το ποιο είναι το φύλλο που τραβήχτηκε, καθώς και ποιο είναι το νέο συνολικό άθροισμα.
δ) Μία συνάρτηση που να εμφανίζει τον νικητή του παιχνιδιού (αν υπάρχει) ή κατάλληλο μήνυμα για την έκβαση του παιχνιδιού.

Συμβουλευτείτε τα παρακάτω παραδείγματα εξόδου, ώστε να δείτε τα μηνύματα που πρέπει να εμφανίζει το πρόγραμμά σας:

Παράδειγμα 1:

First card of dealer is 3
First card of player is 'J'
Second card of player is 4
Player has total 14
Hand_1: Do you want to draw another card (1:for yes)? 0
Dealer draw 'A'
Dealer has total 4 or 14
Dealer draw 'Q'
Dealer has total 14
Dealer draw 9
Dealer has total 23
Player wins!

Παράδειγμα 2:

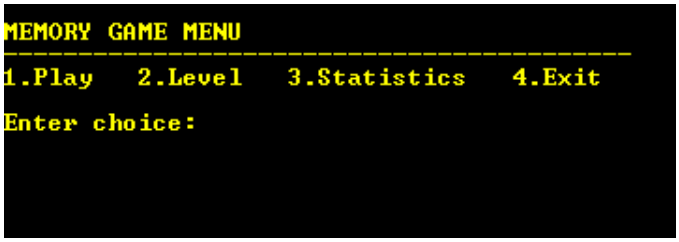
First card of dealer is 8
First card of player is 'J'
Second card of player is 'J'
Split the cards (1:for yes)? 1
Hand_1: Do you want to draw another card (1:for yes)? 1
Player draw 'K'
Player has total 20
Hand_1: Do you want to draw another card (1:for yes)? 0
Hand_2: Do you want to draw another card (1:for yes)? 1
Player draw 8
Player has total 18
Hand_2: Do you want to draw another card (1:for yes)? 0
Dealer draw 7
Dealer has total 15
Dealer draw 3
Dealer has total 18
Hand_1: Player wins!
Hand_2: It's a tie!

Παιχνίδι 3:

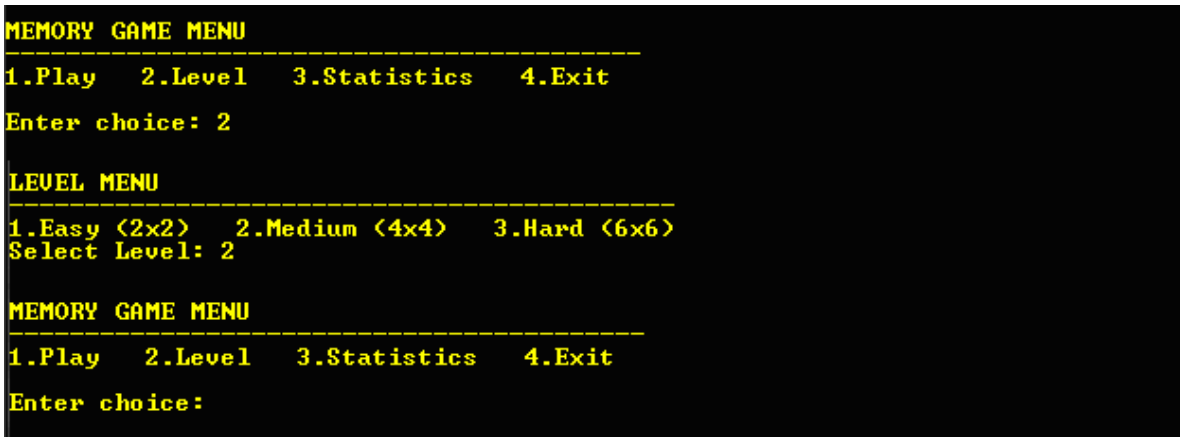
Ένα από τα δημοφιλέστερα επιτραπέζια (πλέον και ηλεκτρονικά) παιχνίδια για παιδιά νηπιακής ή ακόμα και προνηπιακής ηλικίας είναι οι «κάρτες μνήμης». Το παιχνίδι ξεκινά με έναν αριθμό από κάρτες σε ζευγάρια, οι οποίες τοποθετούνται κλειστές (ανάποδα) με τυχαίο τρόπο πάνω σε ένα τραπέζι ή ταμπλό. Ο παίχτης επιλέγει δύο κάρτες τις οποίες και γυρίζει. Αν είναι ίδιες, οι κάρτες παραμένουν ανοιχτές στο ταμπλό μέχρι το τέλος του παιχνιδιού, αλλιώς επανατοποθετούνται στις θέσεις τους κλειστές. Σκοπός του παιχνιδιού είναι να ανοίξουν όλες οι κάρτες (δηλ. ο παίχτης να βρει όλα τα ζευγάρια καρτών) με τις λιγότερες δυνατές κινήσεις.

Γράψτε τις κατάλληλες συναρτήσεις και στη συνέχεια ένα πρόγραμμα που να εξομοιώνει το παιχνίδι «κάρτες μνήμης» βάσει των παρακάτω προδιαγραφών:

i) Θα υπάρχει κεντρικό μενού με τις εξής επιλογές:
`1) Play  2) Level  3) Statistics  4) Exit`
και θα δίνεται η δυνατότητα στον χρήστη να επιλέξει μία από τις ανωτέρω τέσσερις επιλογές.



ii) Με την **επιλογή 2 (`Level`)** θα εμφανίζεται νέο μενού για την τελική επιλογή της δυσκολίας που θέλει να επιλέξει ο παίχτης για το παιχνίδι
[`1. Easy (2x2)  2. Medium (4x4)  3. Hard (6x6)`].
Βάσει της επιλογής αυτής, θα δημιουργείται ένα αντίστοιχο ταμπλό με ζευγάρια καρτών τυχαία τοποθετημένα σε έναν διδιάστατο πίνακα NxN (ή σε έναν μονοδιάστατο πίνακα με NxN στοιχεία που θα «φαίνεται» σαν διδιάστατος), τα οποία θα εξομοιώνονται με ζευγάρια ακεραίων από το 1 έως και το (N*N)/2 και ο χρήστης θα επιστρέφει στο κεντρικό μενού.



Για παράδειγμα, αν ο χρήστης επιλέξει «μέτριο» επίπεδο δυσκολίας (`Medium`), θα εμφανίζεται ένα ταμπλό με τη μορφή 4x4 πίνακα, ο οποίος θα περιέχει τα ζευγάρια των ακεραίων που ανήκουν στο [1, (N*N)/2] για N=4, δηλαδή ζευγάρια των ακεραίων που ανήκουν στο [1, 8], π.χ.:

3 2 8 5
5 3 6 4
1 2 1 7
8 7 4 6

iii) Με την **επιλογή 1 (`Play`)**, αρχίζει το παιχνίδι.

<u>Σημείωση 1:</u> Σε περίπτωση που ο χρήστης δεν έχει νωρίτερα επιλέξει κάποιο επίπεδο δυσκολίας, θεωρούμε ότι το default επίπεδο δυσκολίας του παιχνιδιού θα είναι το «μέτριο» (δηλ. πίνακας 4x4).

<u>Σημείωση 2:</u> Για λόγους testing κατά τη διάρκεια υλοποίησης της εφαρμογής, ενδείκνυται κατά την έναρξη του παιχνιδιού να εμφανίζετε τον πίνακα που θα έχει δημιουργηθεί, ώστε «να έχετε εικόνα» για το πού βρίσκονται τοποθετημένες οι κάρτες και να μην επιλέγετε κάρτες «στα τυφλά».

Κατά την έναρξη του παιχνιδιού, θα εμφανίζεται το board με όλες τις κάρτες «κλειστές» (σε όλες τις θέσεις του board θα φαίνεται ο ίδιος χαρακτήρας, π.χ. "*" ή "?"), ενώ συνίσταται να φαίνονται αριθμημένες και οι γραμμές/στήλες του πίνακα για τη διευκόλυνση του χρήστη.
Στη συνέχεια θα ζητείται από τον χρήστη να δώσει τις «συντεταγμένες» (γραμμή/στήλη) της πρώτης κάρτας που επιθυμεί να ανοίξει. Η τιμή της πρώτης κάρτας, θα πρέπει στη συνέχεια να αναγράφεται σε κατάλληλο μήνυμα και ταυτόχρονα θα ζητείται από τον χρήστη να ανοίξει και τη δεύτερη κάρτα που επιθυμεί, με τον ίδιο ακριβώς τρόπο (γραμμή/στήλη).

```
MEMORY GAME MENU
----------------------------------------
1.Play   2.Level   3.Statistics   4.Exit

Enter choice: 1

First (and last) display of the memo-cards' board (just for testing purposes)
  2  8  7  6
  5  1  4  2
  3  7  8  3
  4  1  6  5


    1 2 3 4
   ---------
1 ¦ * * * *
2 ¦ * * * *
3 ¦ * * * *
4 ¦ * * * *

Please insert the first card row and column (seperated by space).
2 1
Its value is: 5
Please insert the second card row and column (seperated by space).
4 4
```

Αν η δεύτερη κάρτα που επέλεξε ο χρήστης να ανοίξει είναι ίδια με την πρώτη, τότε θα εμφανίζεται σχετικό μήνυμα και οι αντίστοιχες κάρτες θα παραμένουν ανοιχτές για όλη την υπόλοιπη διάρκεια του παιχνιδιού.

```
    1 2 3 4
   ---------
1 ¦ * * * *
2 ¦ 5 * * *
3 ¦ * * * *
4 ¦ * * * 5
Cards Matched!
```

Σε περίπτωση που ο χρήστης δεν μαντέψει σωστά και η δεύτερη κάρτα δεν είναι ίδια με την πρώτη τότε θα εμφανίζεται το board συμπληρωμένο με τις διαφορετικές κάρτες που σήκωσε

```
    1 2 3 4
   ---------
1 ¦ * * * *
2 ¦ 5 * * *
3 ¦ * * * *
4 ¦ * * * 5
Please insert the first card row and column (seperated by space).
1 1
Its value is: 2
Please insert the second card row and column (seperated by space).
3 3
    1 2 3 4
   ---------
1 ¦ 2 * * *
2 ¦ 5 * * *
3 ¦ * * 8 *
4 ¦ * * * 5
```

και στη συνέχεια θα ξανατυπώνεται το board με τις δύο αυτές κάρτες ξανά κλειστές.

```
    1 2 3 4
   ---------
1 ¦ * * * *
2 ¦ 5 * * *
3 ¦ * * * *
4 ¦ * * * 5
Please insert the first card row and column (seperated by space).
```

<u>Σημείωση 3:</u> Κατά τη διαδικασία «ανοίγματος» μιας κάρτας θα πρέπει να γίνεται πάντοτε έλεγχος ώστε ο χρήστης να μην μπορεί να ανοίξει μία ήδη ανοιχτή κάρτα, αλλά – σε αυτή την περίπτωση – ο χρήστης θα πρέπει να ενημερώνεται με κατάλληλο μήνυμα.

<u>Σημείωση 4:</u> Για την τυχαία κατανομή των καρτών στο board να υλοποιηθεί σχετική συνάρτηση, η οποία να χρησιμοποιεί τη συνάρτηση βιβλιοθήκης `rand()`. *Επίσης, για την εμφάνιση του board κάθε φορά στην οθόνη να υλοποιηθεί σχετική συνάρτηση.*

Τέλος, το πρόγραμμα θα συνεχίσει να προτρέπει τον χρήστη να σηκώσει δύο νέες κάρτες ακολουθώντας την ίδια ακριβώς διαδικασία που έχει ήδη περιγραφεί παραπάνω, έως ότου ολοκληρωθεί το παιχνίδι, δηλαδή ο χρήστης βρει όλα τα ζευγάρια των καρτών (σε αυτή την περίπτωση θα εμφανίζεται κατάλληλο μήνυμα ότι το παιχνίδι τελείωσε και ο χρήστης θα οδηγείται αυτόματα στο κεντρικό μενού).

```
Congratulations!!! You won!!!

MEMORY GAME MENU
----------------------------------------
1.Play   2.Level   3.Statistics   4.Exit

Enter choice:
```

iv) Με την **επιλογή 3 (Statistics)**, ο χρήστης θα μπορεί να δει ορισμένα στατιστικά στοιχεία του παιχνιδιού, όπως το επίπεδο δυσκολίας που είχε επιλέξει και σε πόσες συνολικές κινήσεις ολοκλήρωσε το παιχνίδι.

```
MEMORY GAME MENU
----------------------------------------
1.Play   2.Level   3.Statistics   4.Exit

Enter choice: 3

Level: Medium

CONGRATS!!! You've done it after 18 tries!!
```

v) Με την **επιλογή 4 (Exit)**, ο χρήστης θα τερματίζει το παιχνίδι.

<u>Γενική Σημείωση:</u> *Όλες οι συναρτήσεις που θα υλοποιήσετε στο πρόγραμμα θα πρέπει να είναι «γενικές», δηλαδή θα πρέπει να μπορούν να εφαρμόζονται για οποιοδήποτε «επίπεδο δυσκολίας» (Easy, Medium ή Hard). Π.χ. η συνάρτηση που θα τυχαιοποιεί τις κάρτες στο board θα πρέπει να είναι* ***<u>μόνο μία</u>*** *(και όχι τρεις συναρτήσεις, δηλ μία που θα εφαρμόζεται για board* `2x2`*, μία για board* `4x4` *και μία για board* `6x6`*). Το ίδιο θα πρέπει να ισχύει και για την αντίστοιχη συνάρτηση που θα εμφανίζει το board στον χρήστη αλλά και για όλες τις υπόλοιπες συναρτήσεις του προγράμματος. Λύσεις που δεν θα λάβουν υπόψιν τους τη συγκεκριμένη απαίτηση, δεν θα βαθμολογηθούν.*

```
printf("Kales giortes kai kalh xronia!!!\n");
```