Πέμπτη Εργαστηριακή Άσκηση
Διαδικαστικός Προγραμματισμός 2022-2023

Δίνεται ο κάτωθι κώδικας

```
#include <stdio.h>
#include <stdlib.h>
int myand(int a, int b);
int myor(int a, int b);
typedef int (*CallBack)(int, int);
int report(CallBack f);
int main( ) {
	CallBack f ;
	f = myand;
	report(f);
	f = myor;
	report(f);
	return 0;
}
int myand (int a, int b) {
	return a * b;
}
int myor (int a, int b) {
	return a + b>0;
}
int report(CallBack f) {
	printf("%d %d %d\n", 0, 1, f(0, 1));
	return 0;
}
```

Α) Να εξηγήσετε τη λειτουργία του κώδικα. Τι τύπου είναι η μεταβλητή f και τι τιμές παίρνει;

Β) Να τροποποιήσετε τον ανωτέρω κώδικα ώστε η συνάρτηση report να τυπώνει τον πίνακα αληθείας της συνάρτησης την οποία λαμβάνει ως όρισμα. Άρα αν κληθεί ως εξής

```
report(myand);
```

θα πρέπει να τυπώνει

```
0 0 0
0 1 0
1 0 0
1 1 1
```

Γ) Να υλοποιήσετε τις συναρτήσεις `mynand(int, int)`, `mynor(int, int)`, `myxor(int, int)` και να χρησιμοποιήσετε το myfunctions, που θα ορίσετε ως εξής:

```
CallBack myfunctions[] = {myand, myor, myxor, mynand, mynor};
```

για να τυπώσετε τους πίνακες αληθείας όλων των ανωτέρω συναρτήσεων. Τι τύπου είναι το myfunctions; Να επιδιώξετε ο κώδικάς σας να είναι όσο το δυνατό γενικεύσιμος και παραμετρικός. Ενδεικτικά, για να προστεθεί μια επιπλέον συνάρτηση, να αρκεί να προστεθεί (εκτός από τη δήλωση και τον ορισμό της) μόνο το όνομά της στην αρχικοποίηση του myfunctions.

Δ) Να τροποποιήσετε τον κώδικά σας ώστε να τυπώνεται πριν από τον πίνακα αληθείας και ένα αλφαριθμητικό που θα χρησιμοποιήσετε ως όνομα, που θα είναι χαρακτηριστικό της κάθε συνάρτησης. Ο κώδικάς σας να είναι c90, παραμετρικός και γενικεύσιμος. **Για να γίνει αυτό, να ορίσετε και να χρησιμοποιήσετε κατάλληλο πίνακα δομών, όπου κάθε δομή θα έχει ως μέλη έναν κατάλληλο δείκτη σε συνάρτηση και έναν πίνακα χαρακτήρων. Η εκτύπωση του ονόματος θα πρέπει να γίνεται οπωσδήποτε μέσα στην συνάρτηση report.**

Για το σπίτι

1) Να τροποποιήσετε τις συναρτήσεις myand, myor κτλ, ώστε να λαμβάνουν ως είσοδο απλά διασυνδεδεμένη λίστα με στοιχεία τύπου Data και να δείξετε ότι λειτουργούν σωστά γράφοντας κατάλληλο κώδικα στη main(). Δίνονται οι ακόλουθες δηλώσεις:

```
typedef struct data {
      int value;
      struct data * next; } Data;
typedef Data * DataList;
int myandlst(DataList );
Data * createData( int value) {
      Data * dataptr;
      dataptr = malloc(sizeof (Data));
      dataptr->value = value;
      dataptr->next = NULL;
      return dataptr;
}
void appendData(DataList *lstptr, Data *newptr) {
      if (*lstptr==NULL) {
            *lstptr = newptr;
            return;
      }
      appendData( &((*lstptr)->next), newptr);
      return;
}
```

2) Να τροποποιήσετε την συνάρτηση report ώστε να τυπώνει πίνακες αληθείας με τόσες εισόδους όσες δηλώνει ένα δεύτερο όρισμα της τύπου int. Να χρησιμοποιήσετε τις συναρτήσεις που γράψατε στο (Ε).

Για παράδειγμα η κλήση report(myand, 4)να τυπώνει

```
0 0 0 0 0
0 0 0 1 0
0 0 1 0 0
0 0 1 1 0
0 1 0 0 0
0 1 0 1 0
0 1 1 0 0
0 1 1 1 0
1 0 0 0 0
1 0 0 1 0
1 0 1 0 0
1 0 1 1 0
1 1 0 0 0
1 1 0 1 0
1 1 1 0 0
1 1 1 1 1
```

Να δείξετε ότι λειτουργεί σωστά για 2, 3, 4, 5, 6, 8 και 10 εισόδους.

3) Δίνεται ο ακόλουθος κώδικας

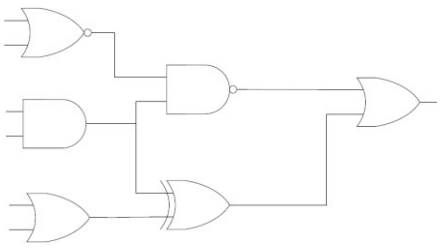
```
typedef struct gate {
	CallBack f;
	struct gate * in1 ;
	struct gate * in2 ; } Gate;

int getinput() {
	int x;
	scanf("%d", &x);
	return x;
}

Gate * creategate(CallBack f) {
	Gate * temp ;
	temp = malloc(sizeof (Gate));
	temp->f = f;
	temp->in1 = NULL;
	temp->in2 = NULL;
	return temp;
}

int eval(Gate *x) {
	int a, b;
	if (x->in1 != NULL)
		a = eval(x->in1);
	if (x->in2 != NULL)
		b = eval(x->in2);
	if (x->in1==NULL && x->in2 == NULL)
		return (x->f)(0,0);
	else
		return (x->f)(a,b);
}
int main( ) {
	Gate * a_ptr, * b_ptr, * c_ptr;
	a_ptr = creategate(myand);
	b_ptr = creategate(getinput);
	c_ptr = creategate(getinput);
	a_ptr->in1 = b_ptr;
	a_ptr->in2 = c_ptr;
	printf("%d", eval(a_ptr));
	return 0;
}
```

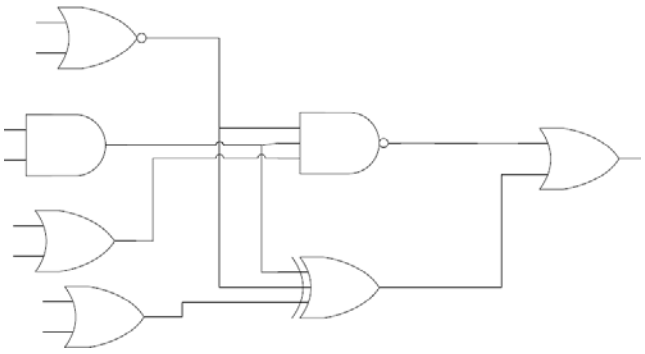
Να τροποποιήσετε τον ανωτέρω κώδικα ώστε να υπολογίζει τις εξόδους του ακόλουθου κυκλώματος, χρησιμοποιώντας κατάλληλους δείκτες τύπου `Gate *`



4) Να τροποποιήσετε τον ανωτέρω κώδικα ώστε να χρησιμοποιεί τον κάτωθι τύπο

```
typedef struct gate {
      CallBack f;
      struct gate ** inputs ;} Gate;
```

όπου inputs είναι αναφορά σε διασυνδεδεμένη λίστα εισόδων. Αυτό επιτρέπει τη σύνδεση οποιουδήποτε αριθμού πυλών. Προσοχή στη δημιουργία της λίστας εισόδων!!!! Το πρόγραμμα να υπολογίζει εξόδους του ακόλουθου κυκλώματος:



5) Να προσθέσετε στον τύπο Gate κατάλληλο μέλος ώστε να αποθηκεύεται ως αλφαριθμητικό κατάλληλο όνομα για κάθε πύλη (Για παράδειγμα or1, and2, or3, το οποίο να εκτυπώνεται κατά τη διάρκεια του υπολογισμού της εξόδου, από τη συνάρτηση eval). Θα πρέπει να τροποποιηθεί και η συνάρτηση creategate ώστε να αρχικοποιεί κατάλληλα την κάθε πύλη.