ΕΡΓΑΣΤΗΡΙΟ 10

1. Γράψτε στο χαρτί τις απαντήσεις. Να χρησιμοποιείτε πράξεις (and, or, xor) **πάνω σε bits** για να:
   a. Υπολογίσετε το πρόσημο ενός ακεραίου (bit testing)

   b. Εντοπίστε αν δύο ακέραιοι αριθμοί έχουν αντίθετα πρόσημα (χωρίς χρησιμοποίηση bit testing).

2. Τι θα τυπώσουν οι εντολές υποθέτοντας ότι i, j και k, είναι μεταβλητές τύπου unsigned short:

```
i = 8; j = 9;
printf("%d", i >> 1 + j >> 1);

i = 1;
printf("%d", i & ~i);

i = 2; j = 1; k = 0;
printf("%d", ~i & j ^ k);

i = 7; j = 8; k = 9;
printf("%d", i ^ j & k);
```

   Hint: Συμβουλευτείτε τον πίνακα προτεραιοτήτων τελεστών στη διαφάνεια 18-14.

3. Χρησιμοποιώντας bitwise operators υλοποιήστε την συνάρτηση:

```
unsigned short swap_bytes(unsigned short i);
```

   Η συνάρτηση επιστρέφει τον αριθμό που προκύπτει από την ανταλλαγή τον δυο bytes του i.
   Παράδειγμα : εάν έχουμε τον αριθμό, 00010010 00110100 μετά την ανταλλαγή ο αριθμός που πρέπει να επιστρέψει η συνάρτηση θα είναι, 00110100 00010010

4. Θεωρήστε ότι είστε υπεύθυνοι για το σχεδιασμό της μορφής αποθήκευσης δεδομένων για ένα παιχνίδι. Πρέπει να αποθηκεύσετε **συμπαγώς** τις ακόλουθες πληροφορίες:

   - Το όνομα του παίκτη που μπορεί να είναι μέχρι 15 χαρακτήρες συν το χαρακτήρα τερματισμού ('\ 0').
   - Το επίπεδο του παίκτη που μπορεί να κυμαίνεται από 0 έως 99.
   - Το αν ο παίκτης παίζει σε κατάσταση ιδιαίτερης σοβαρότητας (hardcore) ή όχι.
   - Η εμπειρία του παίκτη, που μπορεί να κυμαίνεται 0 - 99999999 μονάδες.
   - Το αν ο παίκτης έχει ολοκληρώσει κάθε ένα από τα 5 κύρια quests σε παιχνίδι. Κάθε quest μπορεί να ολοκληρωθεί ανεξάρτητα από την άλλη (π.χ., ένας παίκτης μπορεί να ολοκληρώσει τα quests 1 και 3, αλλά όχι 2, 4, και 5).

Σχεδιάστε ένα struct, `player_record`, που υλοποιεί αυτή τη μορφή αποθήκευσης. Το struct πρέπει να είναι μόνο 24 bytes ή μικρότερο σε μέγεθος. Για να επιτευχθεί αυτό, θα πρέπει να υπολογίσετε τον ελάχιστο αριθμό των bits που απαιτούνται για την αντιπροσώπευση κάθε κομμάτιου των πληροφοριών και τη χρήση bit fields για να περιορίσετε τα πεδία στο εν λόγω δεδομένο μέγεθος. Εάν οποιοδήποτε από τα πεδία μοιάζει να αποτελούν μια συλλογή από booleans, θα πρέπει επίσης να δημιουργήσετε ένα bit-masking σύστημα για να εκπροσωπήσει αυτά booleans συμπαγώς (compactly).

Βεβαιωθείτε ότι το struct είναι πράγματι 24 bytes από την εκτύπωση του μεγέθους του. Επίσης, δώστε ένα παράδειγμα πώς θα κατασκευαστεί ένα στιγμιότυπο (instance) του struct σας με τα ακόλουθα χαρακτηριστικά:

- Name = "xxProfChaosxx"
- Level = 98
- Hardcore mode enabled
- Experience = 84140915
- Completed quests 1, 2, and 4, but not 3 and 5

<u>Σημείωση</u> : Να μην χρησιμοποιηθεί το __attribute__((packed))

5. Χρησιμοποιήστε ένα union για να εκτυπώσετε κάθε byte ενός int. (Hint: Έχετε υπόψη σας το μέγεθος των ints και των άλλων τύπων δεδομένων.)

6. Να υλοποιήσετε σε C, μια απλή δυαδική βάση δεδομένων ή οποία γράφει πέντε εγγραφές της μορφής

```
typedef struct {
     int id;
     char sex;
     char name[10];
} __attribute__((packed)) PERSON;
```

συνεχόμενα σε ένα αρχείο user.db. Τέλος να διαβάζει από το αρχείο όλες τις εγγραφές και τις τυπώνει στην οθόνη. Δείτε μέσω της εντολής hexdump τα περιεχόμενα του αρχείου.

7. Να γράψετε ένα πρόγραμμα το οποίο να δημιουργεί μια απλή συνδεδεμένη λίστα. Ο κάθε κόμβος να έχει ως δεδομένο ένα ακέραιο αριθμό.

```
typedef struct node{
     int value;
     char odd_even;      // 'o'  or 'e'  as values
     struct node *next;
} NODE;
```

Να υλοποιηθούν οι εξής λειτουργίες:

- Εισαγωγή νέου κόμβου στην τελευταία θέση της λίστας.
- Εκτύπωση του δεδομένου κάθε κόμβου.

- Στη main() να εισαχθούν 5 στοιχεία σε μια συνδεδεμένη λίστα και να εκτυπωθεί η λίστα.
- Στη συνέχεια, να γραφτούν σε δυαδικό αρχείο τα στοιχεία της λίστας.
- Στο τέλος, να δημιουργηθεί μια νέα λίστα, αφού διαβαστούν τα στοιχεία από το δυαδικό αρχείο και να εκτυπωθεί
- Προσθέστε το __attribute__((packet)) στο struct. Πια η διάφορα με το προηγούμενο;