שאלה 4

נוסחת CNF נקראת "k-ספיקה" אם קיימת הצבה למשתנים המספקת לפחות k ליטרלים בכל פסוקית. מטרת שאלה זו לכתוב פונקציה מסוג backtracking המכריעה אם נוסחת CNF היא k-ספיקה. לאורך השאלה m יסמן את מספר הפסוקיות, N יסמן את מספר המשתנים ו-k יסמן את רמת הספיקות (שלושת המספרים הללו הם מטיפוס unsigned int, הניחו ש-N מוגדר ע"י **#define**).

הגדרה: הצבה חלקית ***k-מספקת*** נוסחת CNF אם ורק אם בכל פסוקית בנוסחה יש לפחות k ליטרלים שונים אשר מקבלים ערך TRUE.

תזכורת: נוסחת CNF היא מהצורה C_0 AND C_1 AND … AND C_{M-1} כאשר:

- כל C_i נקראת **פסוקית**, והינה מהצורה: $C_i \equiv \left(L_j \text{ OR } L_k \text{ OR } ... \text{ OR } L_t\right)$
- כל L_j נקרא **ליטרל**, והינו משתנה בולאני (x_j) או ההופכי שלו ($!x_j$).

דוגמא לנוסחת CNF:

$$\left(x_1 \text{ OR } x_4\right) \text{ AND } \left(!x_2 \text{ OR } x_0 \text{ OR } !x_3\right) \text{ AND } \left(!x_4\right) \text{ AND } \left(!x_1 \text{ OR } x_3 \text{ OR } x_0 \text{ OR } !x_4\right)$$

הצבה היא קביעת ערך (TRUE או FALSE) למשתנים הבוליאניים. הצבה חלקית משמעותה קביעת ערך לחלק מן המשתנים. הצבה חלקית **מספקת** את הנוסחה אם ורק אם בכל פסוקית, יש לפחות ליטרל אחד אשר מקבל ערך TRUE.

לצורך שמירת נוסחת ה-CNF והצבות חלקיות נעשה שימוש בטיפוס **truth** המוגדר באופן הבא:

```
typedef enum {FALSE, UNSET, TRUE} truth;
```

הקלט לבעיה: מערך **CNF[m][N]** מטיפוס truth כפי שהוגדר בכיתה וקבוע ספיקות **k** מטיפוס **unsigned int**. בנוסף נעשה שימוש במערך **assignment[N]** מטיפוס **truth** לשמירת ההצבה החלקית (כפי שעשינו בכיתה).

הפלט לבעיה: אם קיימת הצבה חלקית k-מספקת אזי יש להדפיסה. (אם יש מספר הצבות כאלו, מספיק להדפיס אחת מהן.) אם לא קיימת הצבה k-מספקת יש להדפיס הודעה האומרת שהנוסחה איננה k-ספיקה.

הקוד של אלגוריתם DPLL שהוצג בכיתה מובא לנוחיותכם.

```
truth DPLLk(truth CNF[][N], truth assignment[], int m,
     int i, int k)
// i denotes the smallest unassigned variable. To use the
// function call DPLL(CNF, assignment, m, 0).
{
  truth current_truth;
  if (i==0 && k_literals_per_clause(CNF, m, k) == FALSE)
    return FALSE; // perform k_lit test only once, when i==0
  current_truth = CNF_SATk(CNF,assignment,m);
  if (current_truth==TRUE)
    return TRUE;
  if (current_truth==FALSE)
    return FALSE;
  assignment[i]=FALSE;
  if (DPLLk(CNF, assignment, m, i+1, k)==TRUE)
    return TRUE;
  assignment[i]=TRUE;
  if (DPLLk(CNF, assignment, m, i+1, k)==TRUE)
    return TRUE;
  assignment[i]=UNSET;
  return FALSE;
}
truth CNF_SATk (truth CNF[][N], truth assignment[], int m,
     int k)
{
  truth t=TRUE;
  int i;
  truth clause_truth;
  for (i=0;i<m;i++){
    clause_truth=clause_SATk(CNF[i], assignment, k);
    if (clause_truth==FALSE)
      return FALSE;
    if (clause_truth==UNSET)
      t=UNSET;
  }
  return t;
}
truth clause_sat (truth clause[], truth assignment[], int n)
{
  int j;
  truth t=FALSE;
  for (j=0;j<n;j++) {
    if (clause[j]!=0) {
      if (assignment[j]==clause[j])
        return TRUE;
      if (assignment[j]==UNSET)
        t=UNSET;
    }
  }
  return t;
}
```

סעיף א – 4 נקודות

הדפיסו את הפלט הרצוי על שני הקלטים הבאים עם קבוע ספיקות k=2:

```
1. CNF[3][3]={{TRUE, UNSET, UNSET}, {TRUE, TRUE, TRUE}, {TRUE, TRUE, FALSE}}
2. CNF[3][3]={{TRUE, FALSE, FALSE}, {TRUE, TRUE,UNSET},{TRUE, UNSET, FALSE}}
```

1. הנוסחה אינה 2-ספיקה מכיוון שבפסוקית הראשונה יש רק ליטרל אחד.
2. הנוסחה המיוצגת במערך היא

(x1v!x2v!x3)^(x1vx2)^(x1v!x3)

ההצבה x1=TRUE, x2=TRUE, x3=FALSE היא 2-מספקת את הנוסחה, ולכן הנוסחה היא 2-ספיקה.

סעיף ב – 6 נקודות

אם קיימת פסוקית ובה פחות מ-k ליטרלים, ברור שהנוסחה איננה k-ספיקה. כתבו קוד לפונקציה הבאה המחזירה ערך FALSE אם קיימת פסוקית בת פחות מ-k ליטרלים ומחזירה ערך TRUE אם בכל פסוקית יש לפחות k ליטרלים.

```
truth k_literals_per_clause(truth CNF[][N], int m, int k)
{
  int i, j;

  for(i=0; i<m; i++) {
    int num literals = 0;
    for(j=0; j<N; j++)
      num literals += (CNF[i][j] != UNSET);
    if(num literals < k)
      return FALSE;
  }

  return TRUE;
}
```

סעיף ג – 6 נקודות

כתבו קוד לפונקציה הבאה הקובעת האם ההצבה החלקית הנתונה במערך **assignment[]** הינה k-מספקת את האילוץ clause שהינו מערך בגודל n.

```
truth clause_SATk(truth clause[], truth assignment[], int k)
{
  int i;
  int num unset=0, num true=0;

  for(i=0; i<N; i++) {
    if(clause[i] == UNSET)
      continue;
    num unset += (assignment[i] != UNSET);
    num true += (clause[i]== assignment[i]);
  }

  if(num true>=k)
    return TRUE;

  return(((num true+num unset)>=k)? UNSET : FALSE);
}
```

סעיף ד – 9 נקודות

כתבו קוד לפתרון בעיית ה-k-ספיקות. יש לעשות שימוש יעיל ככל הניתן בפונקציה שבסעיף ב'. מותר להשתמש בפונקציות שהוגדרו בסעיפים קודמים גם אם לא עניתם על סעיפים אלו. מותר לכתוב את התשובה על גבי ותוך שימוש בקוד שהוצג בכיתה ונתון בעמוד 21.