1) תשובה לשאלה a. במקרה בו יש לכל החבילות את אותו הערך אולם ה slack שלהם שונה – עדיף להשתמש ב edf.

כפי שלמדנו כאשר משתמשים ב bd הוא לא מתייחס ל slack של החבילות בזמן הטיפול אלא רק ל value ולכן האלגוריתם עלול לעבוד בצורה לא יעילה במקרה זה ולזרוק המון חבילות(הוא מכניס לתור לפי ערך ולכולם אותו ערך ולכן עלול לקרות מצב בו בסוף התור יש את ה slack הנמוך ביותר והוא ייזרק ולא יטופל) לעומת זאת ב edf יש עדיפות לטפל ב slack הנמוך ביותר ולכן יפלו פחות חבילות , ומכיוון שלכל החבילות אותו הערך edf יהיה עדיף על bd.

2) תשובה לשאלה b. יש לנו חבילות עם אותו slack אבל value שונה ,

התשובה לשאלה זו היא ההפך מתשובה ל a , מכיוון ש edf לא מתייחס ל value אלא רק ל slack כל האיברים ייכנסו לתור ומכיוון שלכולם אותו slack , ב edf שפועל על פי slack ולא מתייחס ל value לא תתבצע החלפה בין חבילה עם value נמוך לחבילה חדשה עם ערך יותר טוב ולכן יש סיכוי שאם אין מקום בתור למשל המון חבילות ייפלו למרות שיש להן ערך יותר טוב ממה שיש בתור , לעומת זאת bd מטפל לפי ערך ולכן מכיוון שיש אותו slack והוא הולך על הערכים הגדולים ביותר הוא יהיה יעיל יותר.

3) תשובה לשאלה c יש לנו חבילות עם value , slack שונים.
נחלק את הפתרון לשאלה זו למספר מקרים:
תור קטן ו slack נמוך , תור קטן slack גבוה , תור גדול slack נמוך , תור גדול slack גבוה.

מקרה א – תור קטן , slack נמוך : נעדיף להשתמש ב edf ולא ב bd.
מכיוון שה slack נמוך ו bd אינו מתזמן לפי slack המון חבילות יכולות להיזרק(בדומה ל a)מכיוון ש edf מתזמן לפי slack אזי כמות החבילות שתיפול תהיה נמוכה משמעותית מ bd ולכן יהיה יעיל יותר.

מקרה ב – תור קטן ו slack גבוה – במקרה זה נעדיף להשתמש ב bd על פני edf , edf אכן ידאג שכמה שפחות חבילות ייפלו אולם הוא יבצע תחלופה רבה לפי slack ולכן גם יפיל הרבה והוא לא מתייחס לערך , מצד שני bd מתייחס לvalue ואנו יודעים שהslack מספיק גבוה ולכן הסבירות שחבילה תיזרק עם slack=0 היא נמוכה ונעדיף לטפל במקרה זה לפי value שזה אומר אלגוריתם bd.
מקרה ג – תור גדול ו slack נמוך – במקרה זה נעדיף להשתמש ב bd על פני edf , מכיוון שה slack נמוך אנו בכל מקרה נאבד המון פקטות מהתור כי הוא גדול ולא נספיק להתייחס לכולן , ולכן אם ייפלו נעדיף שייפלו עם ה value הנמוך ביותר ולכן נעדיף להשתמש ב bd שהוא מתעדף לפי value וכך נקבל value גבוה יותר.

מקרה ד – תור גדול וslack גבוה – במקרה זה נעדיף שימוש ב edf על פני bd. מכיוון שה slack מספיק גבוה ונרצה לא לאבד חבילות. אולם אם נבצע את אלגוריתם bd יש סיכוי שנאבד יותר חבילות כי כל פעם ייכנס ערך אחר ויכול לדחוף לסוף התור ולכן ייאבדו פקטות בניגוד ל edf שאם התור מספיק גדול וגם ה slack אזי ייאבדו פחות פקטות כי edf הולך לפי slack.

*יש צילומי מסך בקובץ pdf לדוגמאות להרצה לכך.

D.אלגוריתם נוסף לטיפול:
האלגוריתם יתייחס לגודל התור ל value, slack של החבילות.

אם התור מספיק גדול(נניח במקרה שלנו 15) אזי נבדוק לגבי slack שנשאר לחבילות אם יש יותר מ 3 חבילות בתור עם slack<3 נסיר 2 חבילות בכדי שייכנסו חבילות חדשות , כך התור שלנו יתעדכן באופן דינאמי(כל עוד יש עדיין חבילות נכנסות).
ההוצאה מהתור לפעולה לפי ה slack הנמוך ביותר- אם יש יותר מ 1 אז ניקח את האחד עם ה value הגבוה ביותר.
באופן זה האלגוריתם משלב בין bd ל edf כך שהוא עובד לפי slack- דבר שגורם לאופטימיזציה יותר טובה ודואג לכך שפחות חבילות יגיעו ל slack=0 , ומצד שני גם מתייחס ל value.
אם התור קטן נמיין לפי slack , אם יש יותר מאיבר אחד עם slack<3 - נסיר בכדי שלא נתפוס מקום בתור ורוב הסיכוי שלא נטפל בו.

יתרונות וחסרונות אלגוריתמים:
Bd עלול לגרום לנפילה של הרבה חבילות עקב אי התייחסות ל slack, הייתרון שלו שהוא הולך על ה value הגדולים ביותר ולכן במקרה בו למשימות יש slack גבוה הוא יעיל(וגם במקרים שבהם יש הרבה חבילות ולחלק value גבוה).
Edf לא מתייחס ל value אלא רק ל slack ולכן אם יש מספר חבילות עם אותו slack הוא עלול לקחת את החבילה עם ה value הנמוך ושאר החבילות ייפלו.
מצד שני edf פועל כך שכמה שפחות חבילות יפלו= יעילות גבוהה , ולכן גורם למערכת לעבוד חלק.
Edf הינו אופטימלי בין כל קריטריוני העדיפות הדינמית.
האלגוריתם שלי – מסובך לתכנות , לא יעיל בזמן ריצה אמיתית אם יש תור גדול וגם slack , מצד שני בריצה רגילה כמו במקרים שלנו במטלה הוא יכול להיות יעיל יותר בכך שמתייחס גם ל slack וגם ל value.

** אלגוריתם נוסף מוכר הינו אלגוריתם LST.
אלגוריתם זה בדומה ל edf עובד לפי ה slack, אולם לכל תהליך בתור יש גם זמן שצריך על מנת לבצע אותו , וה slack אצלו מוגדר באופן שונה –
Slack='D-t-e'
D- הזמן לטיפול שיש לחבילה.
t- כמה time slots עברו.
e- זמן שנותר לביצוע המשימה.
היתרונות – בדומה ל edf אופטימלי מבחינת cpuוגם ביצוע – רק מקרים בהם יש נתינת "זכות קדימה" לתהליכים. שניתן לשנות קדימויות .
היתרון שלו שמתעדכן תוך כדי ריצה ולכן יודע לעבוד ולייעל כשמגיעות המון חבילות שונות.
חסרון – "לא מסכל קדימה, ז.א. שבזמן שעובד על תהליך מסויים ומטפל רק בו , ולכן במהלך עומס על משאבי מערכת, LST יכול להיות לא אופטימלי., וכמובן זה יהיה כך גם בשימוש בתהליכים בלתי ניתנים להפרעה.