שאלה 1

```
public class BinaryTree
{
    public int SumOfLeavesGreaterThan5(TreeNode root)
    {
        if (root == null)
            return 0;

        // אם זה עלה וערכו גדול מ-5, נחשב אותו לחלק מהסכום
        if (root.Left == null && root.Right == null && root.Value > 5)
            return root.Value;

        // רקורסיה על הילדים של הצומת הנוכחי
        return SumOfLeavesGreaterThan5(root.Left) + SumOfLeavesGreaterThan5(root.Right);
    }
}

class Program
{
    static void Main(string[] args)
    {
        // דוגמה לשימוש בפונקציה
        TreeNode root = new TreeNode(10);
        root.Left = new TreeNode(5);
        root.Right = new TreeNode(8);
        root.Left.Left = new TreeNode(12);
        root.Left.Right = new TreeNode(3);
        root.Right.Left = new TreeNode(6);
        root.Right.Right = new TreeNode(7);

        BinaryTree tree = new BinaryTree();
        int sum = tree.SumOfLeavesGreaterThan5(root);
        Console.WriteLine("Sum of leaves greater than 5: " + sum);
    }
}
```

שאלה 2

```
public static Stack<string> GetStringsByLength(string[] strings)
 {
     Stack<string> stack = new Stack<string>();

     foreach (string str in strings)
     {
         // בדיקה אם אורך המחרוזת תואם את התנאי
         if (str.Length == 'זו גי'.Length)
         {
             // הוספת המחרוזת למחסנית
             stack.Push(str);
         }
     }

     return stack;
 }
```

```
public static void Main(string[] args)
{
    string[] strings = { "זו גי מחרוזת אחת", "זו גי מחרוזת שנייה", "מחרוזת תקינה", "עוד מחרוזת כאן" };

    // קריאה לפונקציה והדפסת התוצאה
    Stack<string> result = GetStringsByLength(strings);
    Console.WriteLine("Strings with length of 'זו גי':");
    while (result.Count > 0)
    {
        Console.WriteLine(result.Pop());
    }
}
```

שאלה 3

א. כן, ישנן שורות במקום הגרסה של #C שבהן יתרחשו שגיאות קומפילציה:

1. שורה 11: בממשק Ix מוגדרת פונקציה חסרת גוף שמכונה ,sod אך במחלקה A לא מוגדר פונקציה ,sod לכן נקבל שגיאת קומפילציה שהפונקציה sod במחלקה A אינה מממשת את הפונקציה sod שמוגדרת בממשק .Ix ניתן לתקן זאת על ידי הוספת מימוש לפונקציה sod במחלקה.A

ב. כן, ישנן שורות שמהוות העמסת פונקציות:

1. שורה 20 במחלקה B מחודשת את הפונקציה foo שנמצאת במחלקה.A

ג. כן, ישנן שורות שמהוות דריסת פונקציות:

1. שורה 20 במחלקה B מדריסה את הפונקציה foo שנמצאת במחלקה.A

ד. לגבי הופעת:()aa.sod

המקור של sod במחלקה A יופעל, לכן ההדפסה תהיה:

A.sod

A.foo 5

ה. לגבי הופעת:()ab.sod

המקור של sod במחלקה A יופעל, אך מכיוון ש abהוא מחלקה מסוג ,B ומחלקה B מדריסה את הפונקציה ,foo ההדפסה תהיה:

A.sod

B.foo 5

ו. לגבי הופעת:()xb.sod

כאשר xb הוא מסוג Ix ונופעל ,sod מקור sod שיחקיל אותו כפי שהוא במחלקה ,B מכיוון שהוא מסוג .Bהדפסה:

A.sod

B.foo 5

ז. לגבי הופעת:()xb.foo

מכיוון ש xbמסוג ,lx אין לו גישה לפונקציה foo שמוגדרת במחלקה .B ייקרא כאן שגיאת ריצה.

שאלה 4

ב. עבור כל אחת משורות 11-1:

1. שגיאת זמן ריצה - כאשר משומש אובייקט מסוג D כאובייקט מסוג ,A אין אפשרות לגשת לאובייקט x שבמחלקה D מבחינת גישה, מכיוון שהוא פרטי.
2. תקין - ידפיס."A::36"
3. תקין - ייקרא בניגוד לקונסטרקטור של A וידפיס."A::7 B::7"
4. תקין - ייקרא בניגוד לקונסטרקטור של A וידפיס."A::7 B::7"
5. תקין - ייקרא בניגוד לקונסטרקטור של A וידפיס."A::12 B::12"
6. תקין - ייקרא בניגוד לקונסטרקטור של A וידפיס."A::15"
7. תקין - ייקרא בניגוד לקונסטרקטור של A וידפיס."A::15"
8. שגיאת ריצה - בשורה זו יש להמיר את bb1 למחלקה C כדי להתבצע קריאה לפונקציה.Sod
9. תקין - ייקרא."B...B"
10. תקין - ייקרא."C...C"
11. תקין - ייקרא "..... sh sh sh" ואחרי כך."C...C"

שאלה 5

א

```
public class D
{
   private Node<A> x;

   public int MaxAValue()
   {
      if (x == null)
      {
         throw new InvalidOperationException("The list is empty");
      }

      Node<A> current = x;
      int max = current.Data.GetA();

      while (current.Next != null)
      {
         current = current.Next;
         int currentValue = current.Data.GetA();
         if (currentValue > max)
         {
```

```
            max = currentValue;
        }
    }

    return max;
  }
}
```

ב

```
public void SortMe()
{
  Stack<D> tempStack = new Stack<D>();

  while (dd.Count > 0)
  {
    int maxAValue = dd.Peek().MaxAValue();
    D maxAItem = null;

    foreach (D item in dd)
    {
      if (item.MaxAValue() == maxAValue)
      {
        maxAItem = item;
        break;
      }
    }

    tempStack.Push(maxAItem);
    dd.Pop();
  }

  // Return the sorted stack to the original stack
  while (tempStack.Count > 0)
  {
    dd.Push(tempStack.Pop());
  }
}
```

ג

```
public void AddItem(D d)
{
  Stack<D> tempStack = new Stack<D>();

  while (dd.Count > 0 && dd.Peek().MaxAValue() > d.MaxAValue())
  {
    tempStack.Push(dd.Pop());
  }

  dd.Push(d);

  while (tempStack.Count > 0)
  {
```

```
        dd.Push(tempStack.Pop());
    }
}
```

ד. סיבוכיות הפתרון בשאלה ג תלויה במספר הפריטים במחסנית. כל פעולה של הוספה או הסרה שתתבצע במחסנית תיקח זמן של O(n), כאשר n הוא מספר הפריטים במחסנית. בגלל שאנו עוברים על המחסנית כדי למיין אותה על פי הערך המקסימלי של a בכל איבר, הסיבוכיות הכוללת של הפתרון היא O(n^2).

שאלה 6

א

```
public int CalculateGradeDistance(StudentNode head, string gradeToFind)
{
    int distance = 0;
    StudentNode current = head;

    // Iterate through the linked list until the end or until finding the grade
    while (current != null && current.Student.GetGrade() != gradeToFind)
    {
        distance++;
        current = current.Next;
    }

    // If the grade is found, return the distance
    if (current != null)
    {
        return distance;
    }
    else
    {
        return 0; // If the grade doesn't appear in the list
    }
}
```

ב

```
public GradeStatNode CreateMaxDistanceChain(StudentNode head)
{
    if (head == null)
    {
        return null; // If the list is empty
    }

    // Initialize variables to keep track of maximum distance for each grade
    Dictionary<int, int> maxDistances = new Dictionary<int, int>();
```

```
    // Iterate through the linked list to find maximum distance for each grade
    StudentNode current = head;
    while (current != null)
    {
        int grade = current.Student.GetGrade();
        if (!maxDistances.ContainsKey(grade))
        {
            maxDistances[grade] = CalculateGradeDistance(current, grade);
        }
        else
        {
            int currentDistance = CalculateGradeDistance(current, grade);
            if (currentDistance > maxDistances[grade])
            {
                maxDistances[grade] = currentDistance;
            }
        }
        current = current.Next;
    }

    // Create a new linked list with GradeStat nodes containing the maximum distances
    GradeStatNode newHead = null;
    foreach (var kvp in maxDistances)
    {
        GradeStatNode newNode = new GradeStatNode();
        newNode.GradeStat.SetGrade(kvp.Key);
        newNode.GradeStat.SetDist(kvp.Value);
        newNode.Next = newHead;
        newHead = newNode;
    }

    return newHead;
}
```

שאלה 8

א. כן, יש שורה שתגרום לשגיאת קומפילציה. שורה מספר 38. התקן: להוסיף את קישור האב F לפרמטר "s", כך שהמתודה תהיה תקפה עבור F וגם עבור H.

ב. כן, יש שורות שמהוות העמסת פונקציות:

- שורה 34: פונקציית Sod עם פרמטר x.
- שורה 37: פונקציית Why עם פרמטר s.

ג. כן, יש שורות שמהוות דריסת פונקציות:

- שורה 19: דריסה של הפונקציה Why ממחלקה F על ידי G.
- שורה 43: דריסה של הפונקציה GetX ממחלקה F על ידי H.

בא. הפקודה F fg = new G(9); תדפיס:

in F with 9

in G with 9

ב. הפקודה F fh = new H(); תדפיס:

in F with 5

in H

in F with 7

in G with 7

ג. הפקודה fg.Why(); תדפיס:

G.why

ד. הפקודה fh.Why(); תדפיס:

H.why

ה. הפקודה fh.Sod(3); תדפיס:

H.sod 3

ו. הפקודה fg.GetX(); תדפיס:

7

שאלה 9

```
class Queue
{
   private List<Request> requests;

   public Queue()
   {
      requests = new List<Request>();
   }

   public void Enqueue(Request request)
   {
      requests.Add(request);
   }
}
```

ג

```
class DailyArchive
{
   private List<Request> requests;

   public DailyArchive()
```

```
    {
        requests = new List<Request>();
    }

    public void AddRequest(Request request)
    {
        requests.Add(request);
    }

    // Additional methods as needed...
}

class Archive
{
    private List<DailyArchive> daily_archives;

    public Archive()
    {
        daily_archives = new List<DailyArchive>();
    }

    public void AddToArchive(Request request)
    {
        // Get today's date
        DateTime today = DateTime.Today;

        // Get the corresponding daily archive
        DailyArchive currentArchive = daily_archives.FirstOrDefault(a => a.Date == today);

        // If today's archive doesn't exist, create it
        if (currentArchive == null)
        {
            currentArchive = new DailyArchive();
            daily_archives.Add(currentArchive);
        }

        // Add the request to today's archive
        currentArchive.AddRequest(request);
    }
}
```

```
// Assuming requests are processed daily, we archive yesterday's requests.
Archive archive = new Archive();
DateTime yesterday = DateTime.Today.AddDays(-1);

// Retrieve yesterday's archive
DailyArchive yesterdayArchive = archive.daily_archives.FirstOrDefault(a => a.Date == yesterday);

// If yesterday's archive doesn't exist, create it
if (yesterdayArchive == null)
{
    yesterdayArchive = new DailyArchive();
```

```
    archive.daily_archives.Add(yesterdayArchive);
}

// Add yesterday's requests to the archive
foreach (Request request in yesterdayRequests)
{
    yesterdayArchive.AddRequest(request);
}
```

שאלה 10

א

```
Stack<M> stack = new Stack<M>();
```

ב

```
public void PrintStringMe(Stack<M> stack)
{
    foreach (M item in stack)
    {
        if (item is N || item is L)
        {
            Console.WriteLine(item.StringMe());
        }
    }
}
```

ג. הדפסת הערך של StringMe עבור כל אחד מסוגי האובייקטים:M, N, L, K

- עבור :M הדפסה של התוc.
- עבור :N הדפסה של מספר השלם הנתון בפרמטרx.
- עבור :L הדפסה של תו c פעמיים מופרד בסימן.::
- עבור :K לא ניתן להפעיל פעולת StringMe על אובייקט מסוג K מבלי ליצור שגיאה בזמן ריצה, כיוון שאין פעולה זו מוגדרת במחלקה M או במחלקה.L

ד

```
public void FilterStack(Stack<M> stack)
{
    Stack<M> tempStack = new Stack<M>();

    foreach (M item in stack)
    {
        if (item is K)
        {
            tempStack.Push(item);
        }
```

```
    }

    stack.Clear();

    foreach (M item in tempStack)
    {
        stack.Push(item);
    }
}
```