AUFGABEN (DEUTSCH)

Aufgabe 1 (2 PUNKTE): Entwerfen und implementieren Sie einen ADT Stack, der Integer-Werte speichert. Ihre Stack-Implementierung muss auf verknüpften Listen basieren, was bedeutet, dass die tatsächlichen Daten Ihres ADT Stack in einer verknüpften Liste und nicht in einem Array gespeichert werden müssen. Um Ihre Implementierung zu präsentieren, führen Sie Ihren Code aus und zeigen Sie das Ergebnis der folgenden Operationen nacheinander an (HINWEIS: Ich stelle Ihnen ein Beispiel main.cpp zur Verfügung, aber Sie können auch Ihr eigenes erstellen):

1. Erstellen Sie ein Stack
2. Geben Sie alle Elemente des Stacks und ihre Größe aus
3. Fügen Sie Elementen mit den Werten 100, 200 und 300 zum Stack hinzu
4. Geben Sie alle Elemente des Stacks und ihre Größe aus
5. Entfernen Sie alle Elemente vom Stapel, und geben Sie nach dem Entfernen eines Elements die verbleibenden Elemente auf dem Stapel und ihre Größe aus
6. Lesen Sie das aktuelle Element auf dem Stack und dann geben Sie es aus
7. Fügen Sie ein Element mit dem Wert 400 zum Stack hinzu
8. Lesen Sie das aktuelle Element auf dem Stack und dann geben Sie es aus
9. Geben Sie alle Elemente des Stacks und ihre Größe aus
10. Entfernen Sie ein Element vom Stack
11. Geben Sie alle Elemente des Stacks und ihre Größe aus

Reichen Sie den C++-Code (cpp-Dateien, h-Dateien und auch ein Makefile) für diese Anwendung zusammen mit einem Bildschirmfoto der Ausgabe der oben genannten Operationen ein.

Aufgabe 2 (3 PUNKTE): Entwerfen und implementieren Sie ein Binary Tree ADT in C++. Sie sollten mindestens die folgenden Baumoperationen implementieren: Suchen, Einfügen, Löschen und Drucken in der richtigen Reihenfolge. Um Ihre Implementierung zu demonstrieren, führen Sie Ihren Code aus und zeigen Sie das Ergebnis der folgenden Operationen in der richtigen Reihenfolge an (HINWEIS: Ich stelle Ihnen ein Beispiel main.cpp zur Verfügung, aber Sie können auch Ihr eigenes erstellen):

1. Erstellen Sie ein Binärbaum
2. Drucken Sie alle Elemente des Baums und seine Größe aus
3. Fügen Sie die Elemente 7, 5, 3, 15, 13, 17, 12, 14, 16, 19 in genau dieser Reihenfolge zum Baum hinzu
4. Drucken Sie die Elemente des Baumes in der pre-order Reihenfolge aus
5. Entfernen Sie Element 7
6. Drucken Sie die Elemente des Baumes in der pre-order Reihenfolge aus
7. Entfernen Sie Element 17
8. Drucken Sie die Elemente des Baumes in der pre-order Reihenfolge aus
9. Entfernen Sie Element 15
10. Drucken Sie die Elemente des Baumes in der pre-order Reihenfolge aus
11. Drucken Sie die Elemente des Baumes in der post-order Reihenfolge aus
12. Drucken Sie die Elemente des Baumes in der in-order Reihenfolge aus

Reichen Sie den C++-Code (cpp-Dateien, h-Dateien und auch ein Makefile) für diese Anwendung zusammen mit einem Bildschirmfoto der Ausgabe der oben genannten Operationen ein.

TASKS (ENGLISH TEXT)

Task 1 (2 POINTS): Design and implement an ADT Stack that stores integer values. Your stack implementation must be based on linked lists, meaning that the actual data of your ADT Stack are stored in a linked list and not an array. To test your implementation run your code and show the result of following operations in sequence (NOTE: I provide you with an example main.cpp but you can create your own):

1. Create a stack
2. Print out all the elements of the stack and its size
3. Add elements with values 100, 200, and 300 to the stack
4. Print out all the elements of the stack and its size
5. Remove all elements from the stack, and after you remove an element print out the remaining elements on the stack and its size
6. Peek the current element on the stack and print it out
7. Add element with values 400 to the stack
8. Peek the current element on the stack and print it out
9. Print out all the elements of the stack and its size
10. Remove an element from the stack
11. Print out all the elements of the stack and its size

Submit the C++ code (cpp files, h files, and also a makefile) for this application along with a screen shot of the output of the above operations.

Task 2 (3 POINTS): Design and implement a Binary Tree ADT in C++. You should implement at least the following tree operations: search, insert, delete, and print in pre-order sequence. To showcase your implementation, run your code and show the result of following operations in sequence (NOTE: I provide you with an example main.cpp but you can create your own):

1. Create a Binary Tree
2. Print out all the elements of the tree and its size
3. Add elements 7, 5, 3, 15, 13, 17, 12, 14, 16, 19 in exactly that sequence
4. Print out the elements of the tree in pre-order sequence
5. Remove element 7
6. Print out the elements of the tree in pre-order sequence
7. Remove element 17
8. Print out the elements of the tree in pre-order sequence
9. Remove element 15
10. Print out the elements of the tree in pre-order sequence
11. Print out the elements of the tree in post-order sequence
12. Print out the elements of the tree in in-order sequence

Submit the C++ code (cpp files, h files, and also a makefile) for this application along with a screen shot of the output of the above operations.