Utwórz polecenie lub zestaw poleceń wykonujących przedstawione tutaj zadania. Przyjmij założenie, że operacje są przeprowadzane w funkcji main() (stąd brak konieczności używania zmiennych wskaźników z adresami) i że wykorzystywane są następujące definicje:

```
struct gradeNode {
char lastName[ 20 ];
double grade;
struct gradeNode *nextPtr;
};
typedef struct gradeNode GradeNode;
typedef GradeNode *GradeNodePtr;
```

a) Utworzenie wskaźnika prowadzącego na początek listy o nazwie `startPtr`. Ta lista jest pusta.
b) Utworzenie nowego węzła typu `GradeNode`, do którego prowadzi wskaźnik `newPtr` typu `GradeNodePtr`. Należy przypisać ciąg tekstowy "Janowski" elementowi składowemu `lastName` i wartość 91.5 elementowi składowemu `grade` (użyj funkcji `strcpy()`). Zdefiniuj wszystkie niezbędne deklaracje i polecenia.
c) Przyjmij założenie, że lista, do której prowadzi `startPtr`, składa się z dwóch węzłów — zawierają one po jednym ciągu tekstowym "Janowski" i "Szczepaniak". Te węzły są ułożone w kolejności alfabetycznej. Zdefiniuj polecenia niezbędne do wstawienia w prawidłowej kolejności węzłów z następującymi danymi dla elementów składowych `lastName` i `grade`:
"Adamczyk" 85.0
"Taczyński" 73.5
"Przybylski" 66.5
Podczas operacji wstawiania wykorzystaj wskaźniki `previousPtr`, `currentPtr` i `newPtr`. Wyświetl wartości `previousPtr` i `currentPtr` przed poszczególnymi operacjami wstawienia. Przyjmij założenie, że `newPtr` zawsze prowadzi do nowego węzła, który ma już przypisane dane.
d) Utwórz pętlę `while` wyświetlającą dane poszczególnych węzłów na liście. Do poruszania się po liście wykorzystaj wskaźniki `currentPtr`.
e) Utwórz pętlę `while` usuwającą wszystkie węzły na liście i zwalniającą pamięć zaalokowaną dla poszczególnych węzłów. Do poruszania się po liście i zwalniania pamięci wykorzystaj wskaźniki odpowiednio `currentPtr` i `tempPtr`.