Elementy języka C++: dynamiczne struktury danych – tablica, lista powiązana.

1. **Obsługa łańcuchów o zmiennej długości**
   Stwórz implementację opartą na dynamicznie alokowanych tablicach dla trzech funkcji obsługujących łańcuchy:

   append – funkcja wymaga podania łańcucha i znaku, a w wyniku swojego działania dołącza ten znak do końca łańcucha.

   concatenate – funkcja używa dwóch łańcuchów i dołącza znaki z drugiego do pierwszego.

   characterAt – funkcja wymaga użycia łańcucha oraz odpowiedniej liczby, zwracając znak znajdujący się na wskazywanej przez nią pozycji w łańcuchu (pierwszy znak łańcucha ma indeks równy zeru).

   Napisz kod zakładając, że funkcja characterAt będzie wywoływana często, a dwie pozostałe stosunkowo rzadko. Względna wydajność operacji powinna odzwierciedlać częstotliwość ich stosowania.

2. **Obsługa rejestru studentów o nieznanej liczbie elementów**
   Napisz funkcje przechowujące i modyfikujące kolekcję rejestrów studentów. Pojedynczy rejestr zawiera numer studenta i jego ocenę – dane te są liczbami całkowitymi. Powinny zostać zaimplementowane następujące funkcje:

   addRecord – funkcja używa wskaźnika do kolekcji rejestrów studentów, zawierających numery studentów i ich oceny, aby dodać do niej nowy rejestr z wypełnionymi danymi.

   averageRecord – funkcja wymaga podania wskaźnika do kolekcji rejestrów studentów i zwraca średnią ocen studentów w kolekcji jako liczbę typu *double*.

   Kolekcja może mieć dowolny rozmiar. Operacja addRecord będzie wykonywana często, więc powinna zostać zaimplementowana w wydajny sposób.

3. **Implementacja łańcucha znaków przy użyciu listy powiązanej ***
   Stwórz implementację łańcuchów tekstowych, która zamiast wykorzystywać dynamicznie alokowane tablice, używa listy powiązanej zawierającej znaki. Tak więc będziesz mieć listę powiązaną, w której właściwymi danymi będą pojedyncze znaki. Pozwoli to na operacje powiększania bez potrzeby ponownego tworzenia łańcucha od nowa. Rozpocznij od zaimplementowania funkcji append i characterAt.

4. **Obsługa łańcuchów o zmiennej długości 2**
   Stwórz funkcję substring z następującymi trzema parametrami: zmienną typu arrayString, liczbą całkowitą oznaczającą początkową pozycję oraz liczbą całkowitą określającą długość łańcucha. Funkcja powinna zwrócić wskaźnik do nowo przydzielonego bloku pamięci zawierającego tablicę znaków. Musi ona zawierać znaki pochodzące z oryginalnego łańcucha, poczynając od określonej pozycji i o podanej długości. Pierwotny łańcuch nie powinien zostać zmodyfikowany.

5. **Obsługa łańcuchów o zmiennej długości 3**
   Stwórz funkcję replaceString, która wykorzystuje trzy parametry, każdy o typie arrayString: source, target i replaceText. Funkcja zamienia każde wystąpienie łańcucha target w łańcuchu source łańcuchem replaceText.

6. **Obsługa rejestru studentów o nieznanej liczbie elementów 2**
   Napisz funkcję removeRecord, która używa wskaźnika do struktury studenCollection oraz numeru studenta, a następnie usuwa z kolekcji rejestr z tym właśnie numerem.

7. **Implementacja łańcucha znaków przy użyciu listy powiązanej 2 ***
   Zaimplementuj funkcję concatenate dla implementacji łańcuchów wykorzystującej listę powiązaną. Pamiętaj, że w przypadku wywołania concatenate(s1, s2), w którym oba parametry są wskaźnikami do pierwszych węzłów odpowiednich list powiązanych, funkcja powinna kopiować każdy z węzłów s2 i dołączać go na koniec s1. Oznacza to, że funkcja nie powinna po prostu przypisać polu next ostatniego węzła na liście s1 adresu pierwszego węzła listy s2.

8. **Implementacja łańcucha znaków przy użyciu listy powiązanej 3 ***
   Dodaj funkcję removeChars do implementacji łańcuchów wykorzystującej listę powiązaną. Funkcja powinna usuwać część znaków z łańcucha, korzystając z parametrów określających pozycję i długość. Upewnij się, że pamięć wykorzystywana przez usunięte węzły zostanie prawidłowo zwolniona.

Pytania, a także rozwiązania zadań, można wysyłać na adres: MDABROWSKI@FUW.EDU.PL.