Elektrotehnički fakultet Sarajevo
Odsjek za računarstvo i informatiku
Predmet: Algoritmi i strukture podataka

Pripremna zadaća za Vježbu 2

Cilj vježbe 2 je upoznavanje sa povezanom listom pri čemu će studenti za pripremu uraditi implementaciju liste, a zatim na vježbi modifikovati implementaciju i primijeniti je za rješavanje nekoliko problema

Zadatak 1.

Implementirati apstraktnu generičku klasu Lista koja omogućuje pohranjivanje **proizvoljnog** broja elemenata proizvoljnog (ali istog) tipa. **Veličina liste nije ograničena!** Metode koje treba ponuditi su:

- konstruktor bez parametara, koji kreira praznu listu;
- metodu **brojElemenata()** koja vraća broj elemenata stavljenih u listu (na početku nula); metodu **trenutni()** koja vraća trenutni element u listi; svaka klasa izvedena iz Liste imaće neki atribut (ili više atributa) pomoću kojih pamti koji je element trenutni, ali korisnik taj atribut ne vidi direktno (treba biti privatan); metoda trenutni() treba se moći koristiti i za čitanje i za izmjenu trenutnog elementa (npr. l.trenutni()=13);
- metode **prethodni()** i **sljedeci()** koje pomjeraju trenutni element odnosno mijenjaju vrijednost koju će vratiti **trenutni()**; ako je trenutni element prvi (posljednji), metoda prethodni (sljedeci) ne treba uraditi ništa, ali treba vratiti logičku vrijednost **false,** u suprotnom ako je pomjeranje trenutnog bilo uspješno treba vratiti **true**;
- metode **pocetak()** i **kraj()** koje postavljaju trenutni element na početak odnosno kraj liste;
- metoda **obrisi()** koja briše iz liste trenutni element; nakon brisanja sljedeći element u listi postaje trenutni; ako je obrisan posljednji element u listi, element prije njega treba postati trenutni;
- ako je lista prazna, metode trenutni, prethodni, sljedeci, pocetak, kraj i obrisi trebaju baciti izuzetak;
- metode **dodajIspred(const Tip& el)** i **dodajIza(const Tip& el)** koje dodaju element ispred odnosno iza trenutnog elementa; ako je lista prazna, obje metode dodaju element na prvo mjesto u listi i taj element postaje trenutni element; u suprotnom, nakon dodavanja elementa *ne treba se promijeniti trenutni element*, odnosno trenutni element ostaje isti kao i ranije;
  Primjer:
  ```
  5 3 2 7 (trenutni je 2)
  dodajIspred(6)
  5 3 6 2 7 (trenutni je 2)
  ```
- preklopljeni **operator []** koji omogućuje direktan pristup i-tom članu niza, uključujući njegovo čitanje i izmjenu; u slučaju da je **i** negativan broj ili da je veći ili jednak broju elemenata niza, operator treba baciti izuzetak.

Prilikom implementacije obratite pažnju na sljedeće stvari:

- klasa Lista je ***apstraktna klasa***, nijednu metodu ne treba implementirati niti treba imati atribute;
- u definicijama metoda namjerno nije definisan tip podataka koje metode vraćaju - odredite ga sami; na pitanja o tome nećete dobiti odgovor jer imate dovoljno podataka u tekstu;

- možete u definiciju klase dodati sve ono što je potrebno da pri radu sa izvedenim klasama nikada ne dolazi do curenja memorije (npr. prilikom dodjele, slanja i vraćanja iz funkcija itd.);
- omogućite da se izvedene klase mogu slati u funkciju preko konstantne reference (npr. float dajProsjek(const Lista<float>& l)) i da se nad tom referencom mogu pozivati sve metode koje ima smisla pozivati.

Zadatak 2.

Implementirati klasu NizLista izvedenu iz klase Lista. U ovoj klasi sve nabrojane metode trebaju biti implementirane pomoću običnog C++ niza. Nije dozvoljeno korištenje bibliotečnih klasa (vector, deque, list…).

Zadatak 3.

Implementirati klasu JednostrukaLista izvedenu iz klase Lista. Ovdje sve metode trebaju biti implementirane pomoću *jednostruko povezane dinamički alocirane liste*. Koristiti dinamičku alokaciju strukture Cvor. Također nije dozvoljeno korištenje bibliotečnih klasa.

Napomene.

Pobrinuti se da ne dolazi do curenja memorije, ilegalnog pristupa memoriji (viseći pokazivač) i slično prilikom bilo kakvog rada sa klasama (slanje objekta u funkciju, bez ili sa reference, vraćanje iz funkcije, pridruživanje povratne vrijednosti metode referenci itd. itd.)