

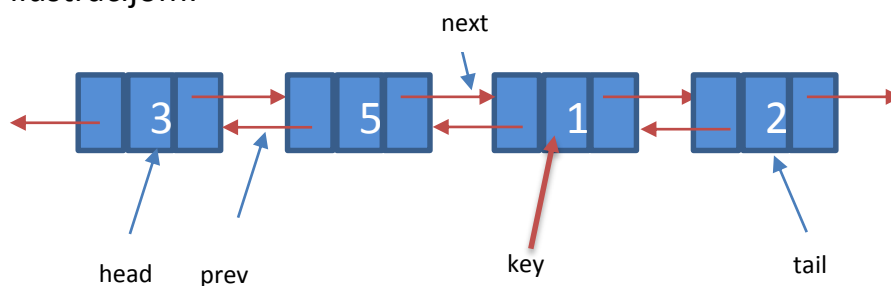
ZADAĆA 3: OSNOVE KLASA

Svaki zadatak nosi određeni broj bodova po podzadatku koji su naznačeni u zagradi. Prvi zadatak treba biti predan kao txt dokument (napravljen u SC5), a ostali kao C++ programski kod. Svi studenti koji predaju zadatke trebaju pristupiti na ScriptRunner5 sustav (<https://mathos.scriptrunner.carnet.hr/>) i tamo objavljivati svoje zadatke u mapi ZADACE->ZADACA_3.

Svi oni studenti koji predaju zadatke moraju biti nazočni na vježbama.

ZADATAK 1 (10):

Dvostruko povezana lista jest lista u kojem svaki element (`Cvor`) se sastoji od neke vrijednosti (`key`), pokazivača na svog prethodnika (`prev`) i pokazivača na sljedbenika (`next`). Lista ima pokazivač na prvi element (`head`) i zadnji element liste (`tail`). Prvi odnosno zadnji element nema prethodnika odnosno sljedbenika pa je njegov pokazivač NIL (`NULL`). Primjer jedne takve liste dano je ilustracijom:



Sljedeća pitanja se tiču razumijevanja upotreba klasa u C++. Za svaki odgovor dajte objašnjenje zašto je točno odnosno netočno.

1. Što je problem u sljedećem programskom kodu i kako bi ga popravili da radi? Što je ispis ovog programa? Odgovor priložite.

<pre> #include <iostream> using namespace std; class Cvor{ public: int n; Cvor next; Cvor prev; public: Cvor(){} Cvor(int x, Cvor* prev, Cvor* next){n = x; next = next; prev = prev; } }; int main() { Cvor x(1,NULL,y); Cvor y(3,x,NULL); Cvor z(5,NULL,x); while(x=x->next) { cout << "<- " << x.n ">-"; } } </pre>	<pre> // korekcija programskog koda /* Ispis programa ... */ </pre>
---	---

ZADATAK 2 (10+10+20):

U programskom jeziku C++ postoji implementacija dvostruko povezane list `list<T>`. Implementirajte dvostruko povezanu listu (vidi definiciju u prethodnom zadatku) danu s prototipom `list` s pripadnim metodama.

```

class node
{
public:
    int value;
    node* _prev;
    node* _next;

    node(){}
    node(int val, node* prev, node* next);

    // pristupi elementima
    int key() const {return val;}
    node* prev() {return _prev;}
    node* next() {return _next;}
};

class list
{
public:
    // konstruktori
    list();
    list(const list&);

    // pristup elementima
    node* front(); // vraća pokazivač na prvi element liste
    node* back();  // vraća pokazivač na zadnji element liste

```

```

void print(); // ispis elemenata liste

// modifikatori liste
void push_front(cvor *x); // dodajte element na početak liste
node* pop_front(); // vraća i briše element s početka liste
void push_back(cvor *x); // na kraj liste stavlja element x
node* pop_back(); // s kraja liste briše i vraća element

void insert(node* prev, node* x); // ubacuje novi čvor x iza čvora prev
void erase(node* x); // briše se čvor x iz liste
void clear(); // metoda briše elemente liste

// operacije
void sort(); // sortira elemente liste po vrijednosti node.value
void reverse(); // obrni redoslijed elemenata u listi

// ostali detalji implementacije su proizvoljni
// ...

};

```

Program testirajte s main funkcijom koja je dana sljedećim programskim kodom:

```

int main()
{
    list L;

    for(int i=0; i<10; i++)
    {
        int val = rand()%10;
        node x(rand()%10, NULL, NULL);
        node y(rand()%10, NULL, NULL);
        L.push_back(x);
        L.push_front(y);
    }

    L.remove(L.front()->next()->next());
    L.print();
    L.sort();
    L.print();
    L.reverse();
    L.print();
    L.clear();

    return 1;
}

```

ZADATAK 3 (40)

Batman i Joker su dugogodišnji neprijatelji. Jedne prilike, Joker je postavio Batmanu zamku tako da je zarobio taoce u zgradi i postavio im bombu. Nažalost, Batman ne zna gdje je bomba, a treba je pronaći i demontirati kako bi spasio taoce. U



pomoć mu dolazi njegov vjerni Alfred koji ga opskrbljuje detektorom topline da može pronaći bombu. Batman može skočiti na prozore zgrade i preko očitavanja detektora topline otkriva da se bomba nalazi negdje u smjeru GORE (U), DOLJE (D), LIJEVO (L), DESNO (R), GORE-LIJEVO (UL), GORE-DESNO (UR), DOLJE-LIJEVO (DL), DOLJE-DESNO (DR). Na temelju te informacije, Batman bira sljedeći položaj da pronađe bombu. Da problem bude interesantniji, Batman ima ograničeni broj skokova koje može koristiti. Pomozite Batmanu da pronađe što brže bombu tako da napravite C++ program koji će mu sugerirati način pronalaženja bombe.

Specifikacija programa:

Implementacija programa se sastoji od 3 klase i jedne procedure:

1. klasa `pozicija` koja služi identifikaciji pozicija na zgradi (mogu biti položaj prozora na zgradi),
2. klasa `Zgrada` koja predstavlja zgradu u kojoj je postavljena bomba
3. klasa `Batman` koja implementira traženje bombe na zgradi
4. Procedura `BatmanVSJoker` koja testira program

Klasa `pozicija` implementira par indeksa kojima se identificiraju prozori na zgradi na kojem Batman može skočiti.

```
class pozicija
{
    public:
        int x,y;
        pozicija()
        pozicija(int x,int y);
        pozicija(const pozicija& p);
};
```

Objekt klase `Zgrada` se instancira preko visine (h) i širine zgrade (w). Možete zgradu promatrati kao pravokutnik, a prozore zgrade kao točke (i, j) $i = 0, 2, \dots, h - 1$ i $j = 0, 2, \dots, w - 1$. Zgrada sadrži zaštićenu varijablu `bomba` u kojem se sprema pozicija Jokerove bombe. U klasi je ponuđena pomoćna metoda `ocitanje(pozicija batman_poz)` koja za danu Batmanu poziciju `batman_poz` vraća jedan od smjerova $\{U, L, D, R, UL, UR, DL, DR\}$ u kojem Batman treba odabrati sljedeću poziciju kako bi došao do bombe.

```
enum smjer {U, L, R, D, UL, UR, DL, DR, None};

class Zgrada
{
    public:
        Zgrada(int w, int h);
        int sirina() const {return w;}
        int visina() const {return h;}

        smjer ocitanje(pozicija batman_poz) const;

    protected:
        pozicija bomba;
        int h, w;
};
```

Klasa `Batman` enkapsulira Batmanovu poziciju i dio zgrade (blok) u kojoj Batman traži bombu. Primijetite da će u startu pretrage blok biti čitava zgrada, a Batmanova početna pozicija je vaš izbor. Konstruktor klase `Batman` instancira objekt preko reference na objekt `Zgrada` i pri tome si definira koordinatni sustav određen objektima klase `pozicija GT, DT`. U konstruktoru se ujedno postavlja početna pozicija Batman-a na zgradi. Metoda `skoci(smjer HeatTrack)` s obzirom na informaciju koje vrati metoda `Zgrada::ocitanje` nalazi novu Batmanovu poziciju na zgradi.

```
class Batman
{
    protected:
        Zgrada zgrada;
        pozicija batman_poz;

        pozicija GT, DT;


    public:
        Batman();
        Batman(const Zgrada& zgrada);
        void skoci(smjer HeatTrack);
```

```
pozicija trenutna_pozicija() const {return batman_poz; }

};
```

Ilustracija primjera zgrade 5x4:

(0,0)			
			Bomba
	Batman		
			(4,5)



Slika: Batman : : GT objekt predstavlja poziciju (0,0), a Batman : : DT objekt predstavlja poziciju (4,5) u startu. Batman je inicijalno postavljen na poziciju (2,4). Metoda Zgrada : : ocitanje vraća vrijednost UR (plava strelica).

Pretraživanje je implementirano u BatmanVSJoker proceduri koja prima visinu i širinu zgrade te broj dozvoljnih skokova te instancira objekte Zgrada i Batman na temelju svog inputa. Unutar beskonačne petlje, procedura računa novu Batmanovu poziciju na temelju vrijednosti Zgrada : : ocitanje. Ukoliko broj skokova koje Batman napravi jest veći od n , onda program završava s porukom 'Batman nije pronašao bombu', inače 'Batman je našao bombu na poziciji (x, y) ' gdje je (x, y) pozicija bombe.

```
void BatmanVSJoker(int h,int w, int n)
{
    Zgrada zgrada(w,h);
    Batman batman(zgrada);

    // broj pokusaja
    int skokovi = 0;

    // implementing search
    while(true)
    {
        /*
        Programski kod koji vi implementirate.

        */

    }
}

int main()
```

```
{  
    BatmanVSJoker(8,4,10);  
    cout << "-----" << endl;  
    BatmanVSJoker(33,425,49);  
    cout << "-----" << endl;  
    BatmanVSJoker(9999,9999,14);  
}
```

Zadatak: Svi navedeni koraci trebaju biti implementirani da se priznaju bodovi.

U ovom zadatku nema parcijalnih bodova.

1. Implementirajte konstruktore klase `Zgrada` i `Batman` koji nasumično postavljaju unutar zgrade bombu odnosno Batmana.
2. Implementirajte metodu `Batman::skoci` koja odabire novu Batmanovu poziciju tako da što prije dođe do bombe.
3. Implementirajte metodu `Zgrada::ocitanje`.
4. Implementirajte proceduru `BatmanVSJoker`.