Muhammed Bedir ULUCAY 1801042637

CSE 241 -501 Midterm Session IV

"I hereby pledge that I will stricly adhere to academic integrity codes and the work done on this examination is solely my own and I will not receive / give any help from/to anybody or source during this examination"

Define and implement a C++ class to represent a vector of doubles vector<double> your new class work with doubles only following

- Uses dynamic memory for storing doubles
- Overladed [] and << ×2
- push_back pop_back, capacity, size > function
- creates a new name spaces
- has separated interface implementation overloaded operator += appends another vector to the end of this vector
- decltype and auto - any other fonctions neded

```
namespaces ulucay {

        class vector {
         public:
            vector();
            vector(int capacity);
            double & operator[](int index);
            double operator[](int index)const;
            friend ostream & operator << (ostream & sout, const vector& obj);
            void push_back(int element);
            void pop-back();
            vector & operator += (const vector& obj);
            vector operator = (const vector & obj); //Big three
            ~vector();
            vector (const vector & obj);
```

②

Muhammed
Bedir
UKUĞAY

```
    private:
        int capacity, size;
        double *dynamic;

};

vector::~vector(){ detete [] dynamic }

vector::vector(const vector & obj)
    : capacity(obj.capacity), size(obj.size) {
    for(auto i=0; i< obj.size; ++i){
        dynamic[i] = obj[i];
    }
}

vector  operator = (const vector & obj) {

    if (this != &obj){
        size = obj.size;
        capacity = obj.capacity;
        double *arr = new double [obj.size];

        for(auto i=0; i< obj.size; ++i)
            arr[i] = obj.dynamic[i];
        detete [] dynamic;
        dynamic = arr;
    }

    return *this;

}
```

Muhammed
Bedir
ULUGAY

```
vector ::vector () : capacity (50), size(0) {
    dyromic = new double[capacity];
}
vector ::vector (int capacity ) : capacity (capacity), size (0) {
    dynamic = new double [capacity];
}
double & vector ::operotor [] (int index) {
    if (index < 0 || index > capacity ) {
        cout << "out of size";
        exit(1);
    }
    return dynomic [index];
}
double vector :: opesetor [] (int index) const {
    if (indx < 0 || index > capacity) {
        cout << "out of size";
        return -1;
    }
    return dynomic [index];
}
ostream & operator<< (astream & sout, const vector & obj ) {
    for (auto i=0 ; i<obj.size; ++i)
        sout << obj[i] << " ";
    sout << "\n";
    return sout;
}
```

4

Muhammed
Bedir
ULUCAY

```
void   vector :: push_back (int element) {
    if ( size == capacity) {
        capacity = capacity * capacity ;
        double *arr = new double[capacity]
        for (int i=0 ; i< size ; ++i)
            arr [i] = dynamic [i];
        delete [] dynamic;
        dynamic = arr;
    }
    dynamic [size] = element;
    size ++;
}
void   vector :: pop_back () {
    dynamic [size -1] = "\0";
    size --;
}
vector &  operator += (const vector & obj) {
    for (auto i=0 ; i< obj.size ; ++i)
        push-back ( obj [i] );
    return *this;
}

} //end of ulucay namespaces
```