```cpp
/*
 * Connor Petri
 * CIS 22C
 * 2020-10-17
 */

template <class T>
class ExtendableArray
{
public:
    // Used a default argument to combine the Default constructor and standard constructor.
    // Arrays of size 0 exist, so Extendable arrays of size 0 should be able to exist too.
    ExtendableArray(unsigned int size = 0)
    {
        this->arr = new T*[size];
        this->arrSize = size;
        this->numElements = 0;
    }

    ExtendableArray(const ExtendableArray &array) // Copy constructor
    {
        this->arr = new T*[array.getSize()];
        this->arrSize, this->numElements = array.getSize();
        for (int i = 0; i < array.getSize(); i++)
        {
            this->arr[i] = array.get(i);
        }
    }

    ~ExtendableArray()
    {
        delete[] this->arr;
    }

    int getSize() { return (int)this->numElements; }

    T * get(unsigned int index)
    {
        if (index >= this->numElements) { return nullptr; }
        return this->arr[index];
    }

    int set(T *ptr, unsigned int index)
    {
        if (index >= this->numElements) { return -1; }
        this->arr[index] = ptr;
        return (int)index;
    }

    int insert(T *ptr, unsigned int index)
    {
        if (index > this->numElements) { return -1; }

        if (++this->numElements > this->arrSize) { realloc(); }

        T *temp1 = nullptr;
        T *temp2 = ptr;
        for (int i = index; i <= this->numElements; i++)
        {
            temp1 = this->arr[i];
            this->arr[i] = temp2;
            temp2 = temp1;
        }

        return (int)index;
    }
    int remove(unsigned int index)
    {
        if (index >= this->numElements) { return -1; }
        for (int i = index; i < this->numElements; i++)
        {
            this->arr[i] = this->arr[i + 1];
        }
        this->arr[this->numElements - 1] = nullptr;
        this->numElements--;

        return (int)index;
    }

    int append(T *ptr)
    {
        return this->insert(ptr, this->numElements);
    }

    int prepend(T *ptr)
    {
        return this->insert(ptr, 0);
    }

protected:
    T **arr;
    unsigned int arrSize;
    unsigned int numElements;

    // when called, realloc() will create a new array of twice the size of the old one and copy
each element over.
    void realloc()
    {
        T **newArr = new T*[this->arrSize * 2];
        for (int i = 0; i < this->arrSize; i++)
        {
            newArr[i] = this->arr[i];
        }
        this->arr = newArr;
        this->arrSize *= 2;
    }
};
/*
 * OUTPUT:
 * Read Array size 100  Read # commands 11
 * Command: A  -1  12345  KleinmanRonald
 * Command: A  -1  87654  SmithMary
 * Command: A  -1  54321  BerraYogi
 * Command: I  1  32145  RizzutoPhil
 * Command: R  2  -1  xxxx
 * Command: A  -1  67890  MantleMickey
 * Command: I  9  89300  KofaxSandy
 * Command: I  3  89012  KofaxSandy
 * Command: A  -1  99887  MarisRoger
 * Command: S  3  62109  FordWhitey
 * Command: I  0  10200  SkowronMoose
 * --------
 *
 * 0  Student: ID = 10200  Name = SkowronMoose
 * 1  Student: ID = 12345  Name = KleinmanRonald
 * 2  Student: ID = 32145  Name = RizzutoPhil
 * 3  Student: ID = 54321  Name = BerraYogi
 * 4  Student: ID = 62109  Name = FordWhitey
 * 5  Student: ID = 67890  Name = MantleMickey
 * 6  Student: ID = 99887  Name = MarisRoger
 *
 * Process finished with exit code 0
 */
```