oj::Vector::Vector()

{

begin\_ = nullptr;

end\_ = nullptr;

last\_ = nullptr;

}

oj::Vector::size\_type oj::Vector::capacity()const

{

if(begin\_ != nullptr){

return (end\_ - begin\_);

}

else

return 0;

}

oj::Vector::size\_type oj::Vector::size()const

{

if(last\_ != nullptr){

return (last\_ - begin\_ )+1;

}

else

return 0;

}

void oj::Vector::pop\_back()

{

if(size() > 1){

last\_ = last\_ - 1;

}

else{

last\_ = nullptr;

}

}

void oj::Vector::push\_back(value\_type val)

{

if(capacity() > size()){

last\_ = begin\_ + size();

\*last\_ = val;

}

else{

reserve(max(capacity()+1, capacity()\*3));

last\_ = last\_ + 1;

\*last\_ = val;

}

}

void oj::Vector::insert(size\_type pos,size\_type count,value\_type val)

{

if(size()+count > capacity() && pos >= size()){

reserve(max(capacity()+capacity()/2, capacity()+count));

for(size\_type i=0; i<count; i++)

push\_back(val);

}

else if(size()+count > capacity() && pos<size()){

reserve(max(capacity()+capacity()/2, capacity()+count));

pointer temp = new oj::Int[size()];

for(size\_type i = pos; i<size(); i++)

temp[i] = begin\_[i];

for(size\_type i = pos; i<pos+count; i++)

begin\_[i] = val;

for(size\_type i = pos+count; i<size()+count; i++)

begin\_[i] = temp[i-count];

delete []temp;

last\_ = last\_ + count;

}

else if(pos >= size()){

for(size\_type i=0; i<count; i++)

push\_back(val);

}

else{

pointer temp = new oj::Int[size()];

for(size\_type i = pos; i<size(); i++)

temp[i] = begin\_[i];

for(size\_type i = pos; i<pos+count; i++)

begin\_[i] = val;

for(size\_type i = pos+count; i<size()+count; i++)

begin\_[i] = temp[i-count];

delete []temp;

last\_ = last\_+count;

}

}

void oj::Vector::reserve(size\_type new\_capacity)

{

if(new\_capacity > capacity()){

size\_type j = size();

end\_ = new oj::Int[new\_capacity];

for(size\_type i = 0; i<j; i++){

end\_[i] = begin\_[i];

}

delete []begin\_;

begin\_ = end\_;

end\_ = begin\_+new\_capacity;

last\_ = begin\_ + j - 1;

}

}

void oj::Vector::resize(size\_type new\_size)

{

if(new\_size > size()){

if(new\_size > capacity()){

reserve(new\_size);

for(size\_type i = size(); i<new\_size; i++)

begin\_[i] = 0;

last\_ = begin\_ + new\_size -1;

}

else{

size\_type i = size();

for(; i<new\_size; i++)

begin\_[i] = 0;

last\_ = begin\_ + new\_size -1;

}

}

else{

if(new\_size == 0)

last\_ = nullptr;

else{

last\_ = begin\_ + new\_size - 1;

}

}

}

oj::Vector::~Vector()

{

delete [] begin\_;

}

size

capacity

push\_back 5

push\_back 4

push\_back 3

push\_back 2

push\_back 1

pop\_back

pop\_back

pop\_back

pop\_back

resize 5

size

capacity

reserve 0

resize 0

insert 0 2 4

0

0

5

5 4

5 4 3

5 4 3 2

5 4 3 2 1

5 4 3 2

5 4 3

5 4

5

5 0 0 0 0

5

9

5 0 0 0 0

4 4

4 4