

NAME : *answers***Open book, open notes, but please do not ask questions.****Write all answers on these sheets.**

question	1	2	3	4	5	total
points						
maximum	20	20	20	20	20	100

1. Write an Abstract Data Type description for complex numbers (numbers of the form  $a + bi$ ,  $i = \sqrt{-1}$ ) where real and imaginary parts can be any number type.

For the operators  $*$  and  $/$ , describe prototype, precondition(s) and postcondition(s).

**Answer:**

```

abstract <typename T> complex;
/* a complex number is a pair of two numbers of type T,
   respectively its real and imaginary part */

abstract operator* ( complex x, complex y );
postcondition: z = x*y is the product of x with y

abstract operator/ ( complex x, complex y );
precondition: y != 0
postcondition: (x/y)*y == x

```

2. Consider the following code:

```
double A[m][n]; // some m-by-n matrix
double x[n];    // a vector of length n
double y[m];    // will store y = A*x
// statements defining A and x are omitted
for(int i=0; i<m; i++)
{
    y[i] = 0.0;
    for(int j=0; j<n; j++)
        y[i] = y[i] + A[i][j]*x[j];
}
```

- (a) Count the number of arithmetical operations in the code above and write the result as a big  $O$  statement in the dimensions  $n$  and  $m$ .

**Answer:**

For every  $i$ , the inner loop controlled by  $j$  is executed  $n$  times.

Every execution of the body of the inner loop involves one addition  $+$  and one multiplication  $*$ . So one execution of the inner loop takes  $n$  additions and  $n$  multiplications, a total of  $2n$  arithmetical operations.

The outer loop controlled by  $i$  is executed  $m$  times. Every time we run through that loop we perform  $2n$  arithmetical operations. So we have a total of  $m \times 2n$  arithmetical operations.

The cost of the code above is  $O(nm)$ .

- (b) Give a loop invariant on the outer loop controlled by  $i$ . Show that satisfying the loop invariant with the negation of the stop condition gives the postcondition.

**Answer:**

Denote by  $y[0..i]$  the first  $i$  elements of  $y$ .

Then the loop invariant, valid for all  $i$  in  $0..m-1$  is:

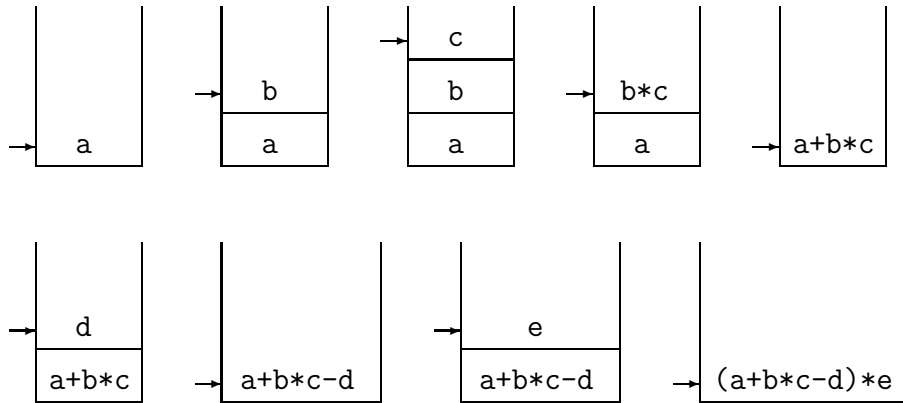
$y[0..i]$  contains the first  $i$  values of the matrix-vector product  $A*x$ .

At  $i == 0$ , the loop invariant applies to zero elements and is therefore true.

At the end, when  $i == m$ ,  $y[0..m] == y$  and then  $y$  equals  $A*x$ .

3. Convert the postfix expression  $a \ b \ c \ * \ + \ d \ - \ e \ *$  to an equivalent infix expression. Show the evolution of the stack.

*Answer:*



/20

4. Given is a STL deque of type `Job`. Write the definition of a function that takes on input a deque and that returns a copy of the deque as a STL list.

- (a) Every element of the deque should occur in the same order in the list.  
 (b) The code should ***not*** change the given deque!

*Answer:*

```
list<Job> copy ( deque<Job> q )
{
    list<Job> L;

    for(deque<Job>::iterator i = q.begin(); i != q.end(); i++)
        L.push_back(*i);

    return L;
}
```

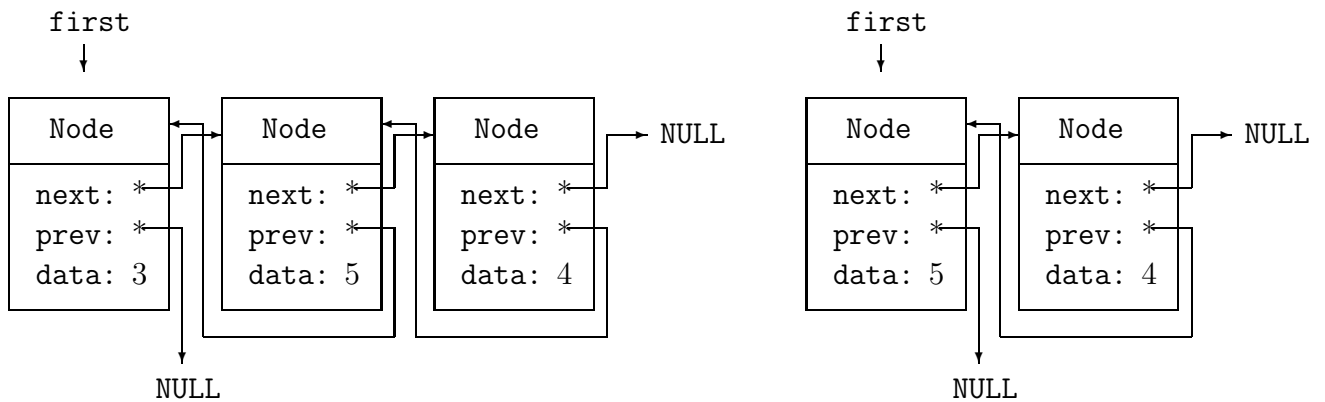
/20

5. Consider the definition of a node (for any type T) in a double linked list:

```
struct Node
{
    T data; // T is template parameter
    Node *next; // pointer to next node
    Node *prev; // pointer to previous node
    Node(const T& item, Node* next_ptr = NULL, Node* prev_ptr = NULL) :
        data(item), next(next_ptr), prev(prev_ptr) {}
};
```

A (linear) list is stored as a pointer with name `first`. This `first` points to the first item in the list and that item is of type `Node`.

(a) Draw a list of a least three elements (use `int` as T) and illustrate the steps to remove the first element of the list. **Answer:** below left is the *before* and right is the *after* the removal of the first element.



(b) Write code for the following function:

```
template <typename T>
List<T>::pop_front()
// removes the first element in the list,
// does nothing if the list is empty
{
```

**Answer:**

```
    if(first != NULL) {
        if(first->next == NULL)
        {
            delete first; first = NULL;
        }
        else
        {
            first = first->next; delete first->prev; first->prev = NULL;
        }
    }
}
```