# COL703: Logic for Computer Science

I semester 2018-19

Study the following screenshots of a bubblesort program written and verified in dafny, along with an executable which has been executed.

Consider your entry number `<entry-number>` which conforms to the regular expression pattern (written in perl-regexp syntax)

```
20[0-9]{2}[A-Z0-5][0-9]{4}
```

Now take the number $n$ formed by the last four digits of your entry number and solve the Problem $m = n\%8$ and submit the verified $m$`.dfy` file and the corresponding $m$`.exe` file as a single `<entry-number>.tgz` file.

Problems are sequentially numbered. All problems are variations of **in-place selection sort** on integer arrays without using any extra arrays and using only **comparisons and swaps** within the given input array. The input array could contain multiple copies of the same element.

Problem 0 (In place) Selection sort (in **non-descending order**) on integer arrays by traversing the unsorted portion of the array from **left to right** and moving the **minimum** element to the **left-most** unsorted place.

Problem 1 (In place) Selection sort (in **non-descending order**) on integer arrays by traversing the unsorted portion of the array from **left to right** and moving the **maximum** element to the **right-most** unsorted place.

Problem 2 (In place) Selection sort (in **non-descending order**) on integer arrays by traversing the unsorted portion of the array from **right to left** and moving the **minimum** element to the **left-most** unsorted place.

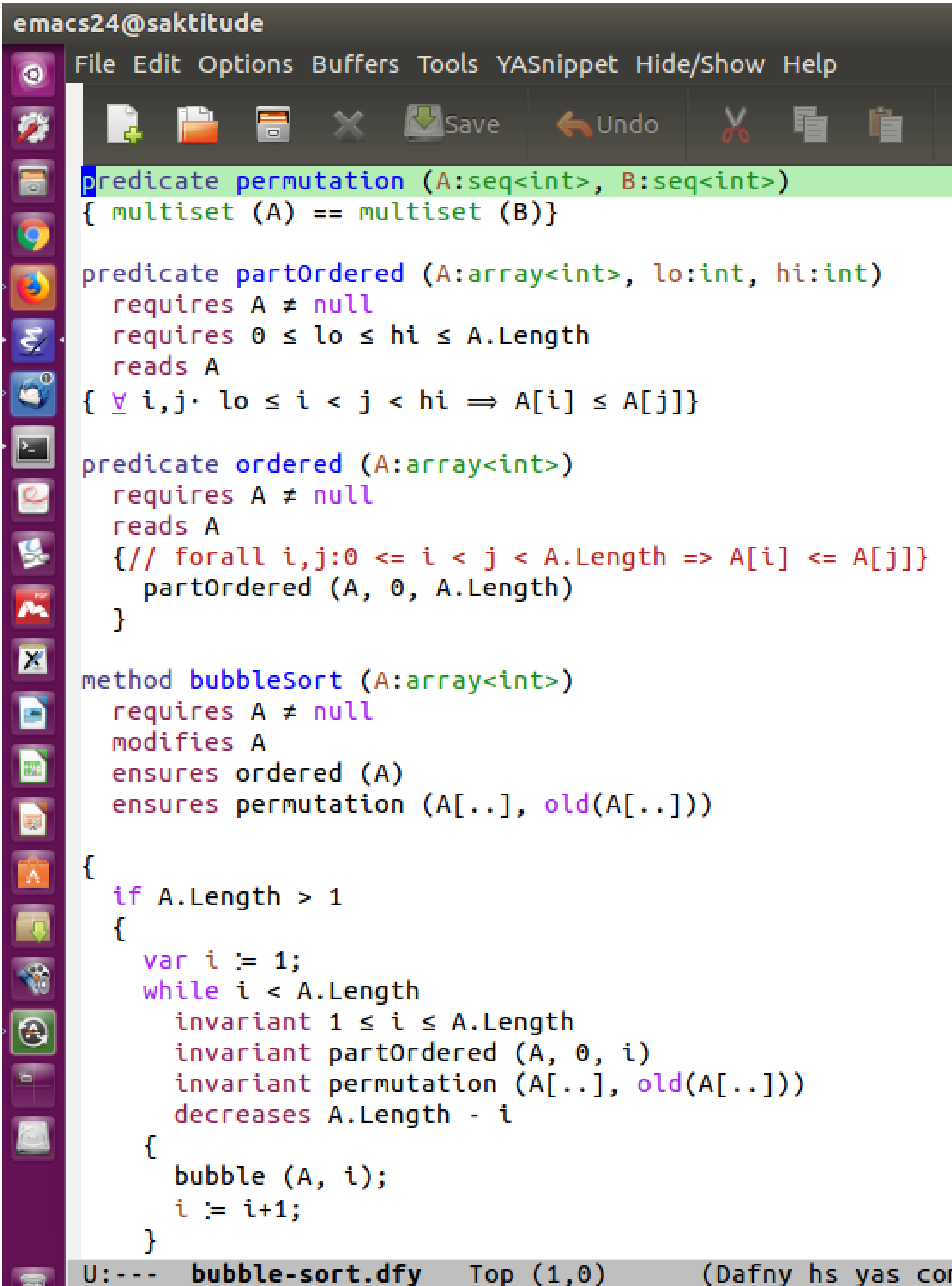
Problem 3 (In place) Selection sort (in **non-descending order**) on integer arrays by traversing the unsorted portion of the array from **right to left** and moving the **maximum** element to the **right-most** unsorted place.

Problem 4 (In place) Selection sort (in **non-ascending order**) on integer arrays by traversing the unsorted portion of the array from **left to right** and moving the **minimum** element to the **right-most** unsorted place.

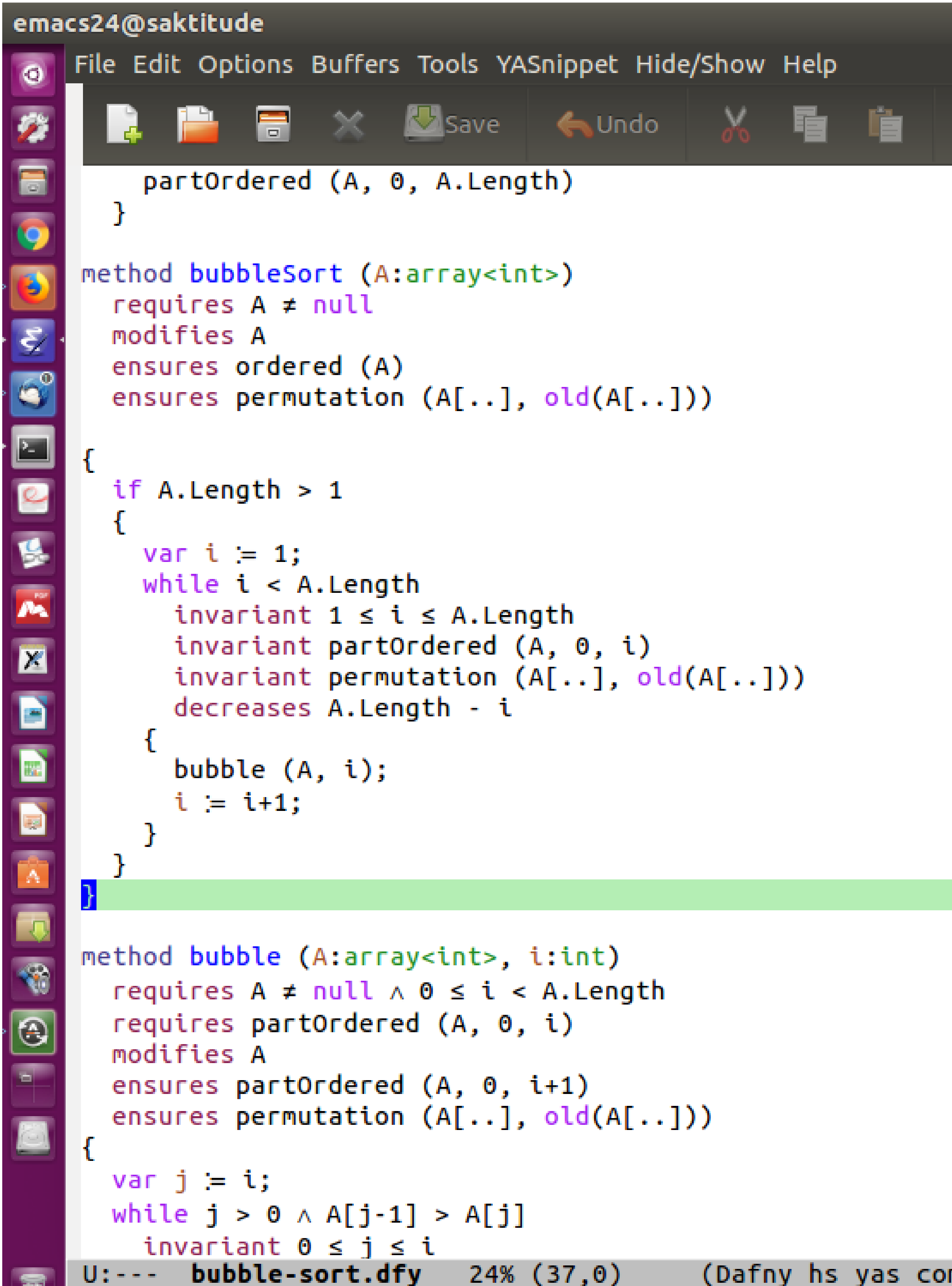
Problem 5 (In place) Selection sort (in **non-ascending order**) on integer arrays by traversing the unsorted portion of the array from **left to right** and moving the **maximum** element to the **left-most** unsorted place.

Problem 6 (In place) Selection sort (in **non-ascending order**) on integer arrays by traversing the unsorted portion of the array from **right to left** and moving the **minimum** element to the **right-most** unsorted place.

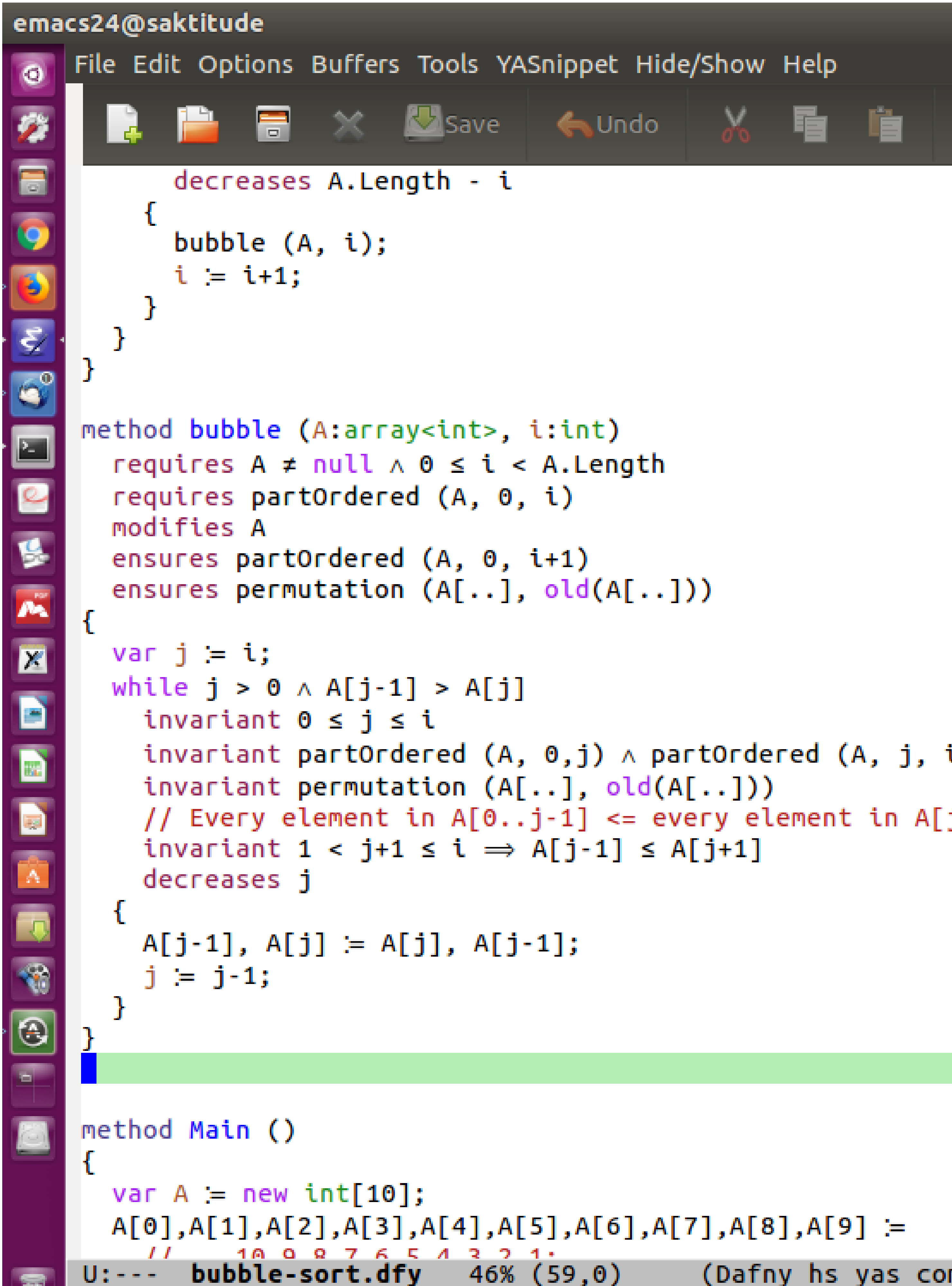
Problem 7 (In place) Selection sort (in **non-ascending order**) on integer arrays by traversing the unsorted portion of the array from **right to left** and moving the **maximum** element to the **left-most** unsorted place.


emacs24@saktitude
File Edit Options Buffers Tools YASnippet Hide/Show Help
Save
Undo
U:---  bubble-sort.dfy    Top (1,0)     (Dafny hs yas co

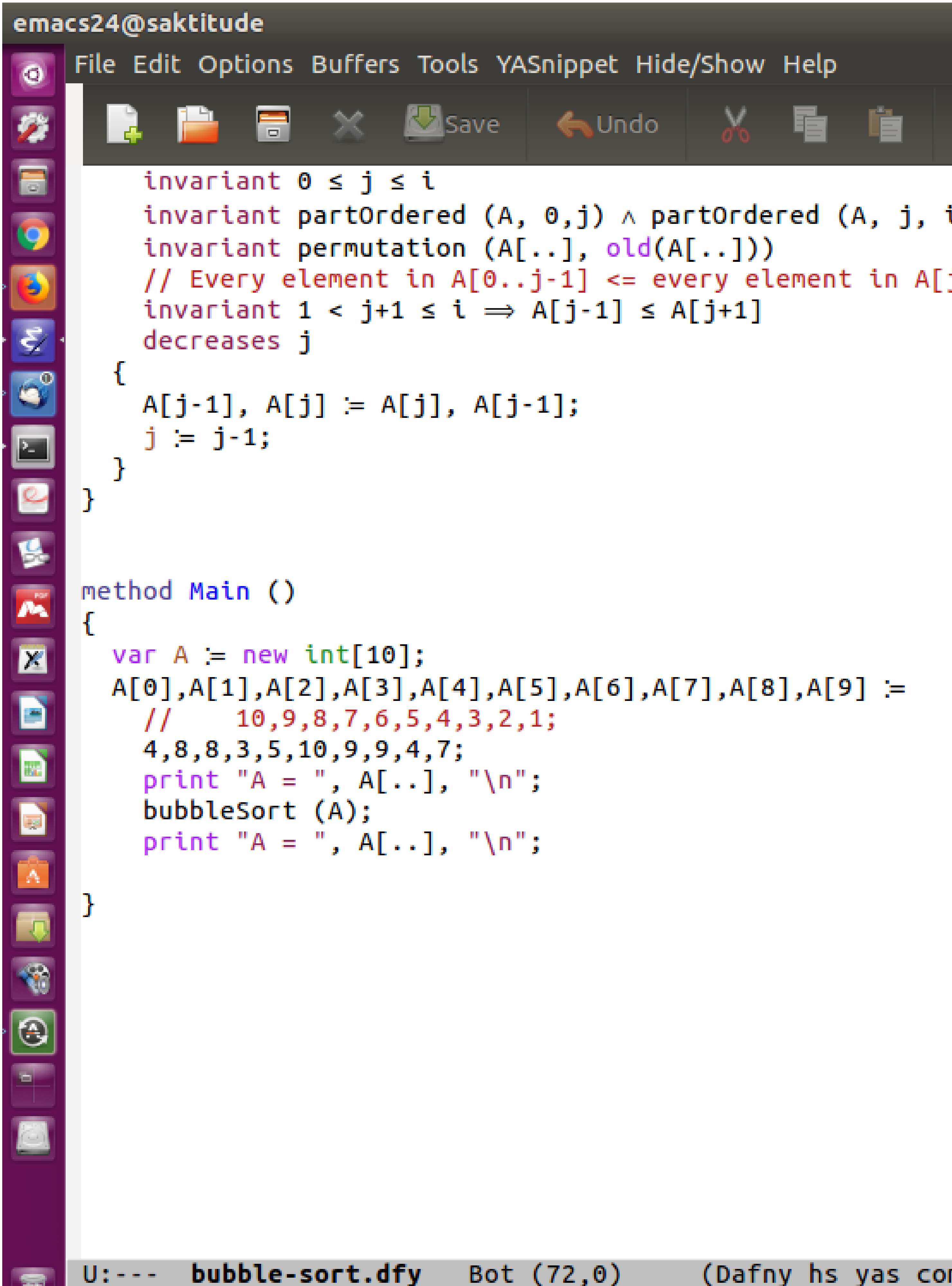
```
predicate permutation (A:seq<int>, B:seq<int>)
{ multiset (A) == multiset (B)}

predicate partOrdered (A:array<int>, lo:int, hi:int)
  requires A ≠ null
  requires 0 ≤ lo ≤ hi ≤ A.Length
  reads A
{ ∀ i,j· lo ≤ i < j < hi ⟹ A[i] ≤ A[j]}

predicate ordered (A:array<int>)
  requires A ≠ null
  reads A
  {// forall i,j:0 <= i < j < A.Length => A[i] <= A[j]}
    partOrdered (A, 0, A.Length)
  }

method bubbleSort (A:array<int>)
  requires A ≠ null
  modifies A
  ensures ordered (A)
  ensures permutation (A[..], old(A[..]))

{
  if A.Length > 1
  {
    var i := 1;
    while i < A.Length
      invariant 1 ≤ i ≤ A.Length
      invariant partOrdered (A, 0, i)
      invariant permutation (A[..], old(A[..]))
      decreases A.Length - i
    {
      bubble (A, i);
      i := i+1;
    }
```


emacs24@saktitude
File Edit Options Buffers Tools YASnippet Hide/Show Help
Save
Undo
U:---  bubble-sort.dfy    24% (37,0)    (Dafny hs yas cor

```
    partOrdered (A, 0, A.Length)
  }

method bubbleSort (A:array<int>)
  requires A ≠ null
  modifies A
  ensures ordered (A)
  ensures permutation (A[..], old(A[..]))

{
  if A.Length > 1
  {
    var i := 1;
    while i < A.Length
      invariant 1 ≤ i ≤ A.Length
      invariant partOrdered (A, 0, i)
      invariant permutation (A[..], old(A[..]))
      decreases A.Length - i
    {
      bubble (A, i);
      i := i+1;
    }
  }
}

method bubble (A:array<int>, i:int)
  requires A ≠ null ∧ 0 ≤ i < A.Length
  requires partOrdered (A, 0, i)
  modifies A
  ensures partOrdered (A, 0, i+1)
  ensures permutation (A[..], old(A[..]))
{
  var j := i;
  while j > 0 ∧ A[j-1] > A[j]
    invariant 0 ≤ j ≤ i
```

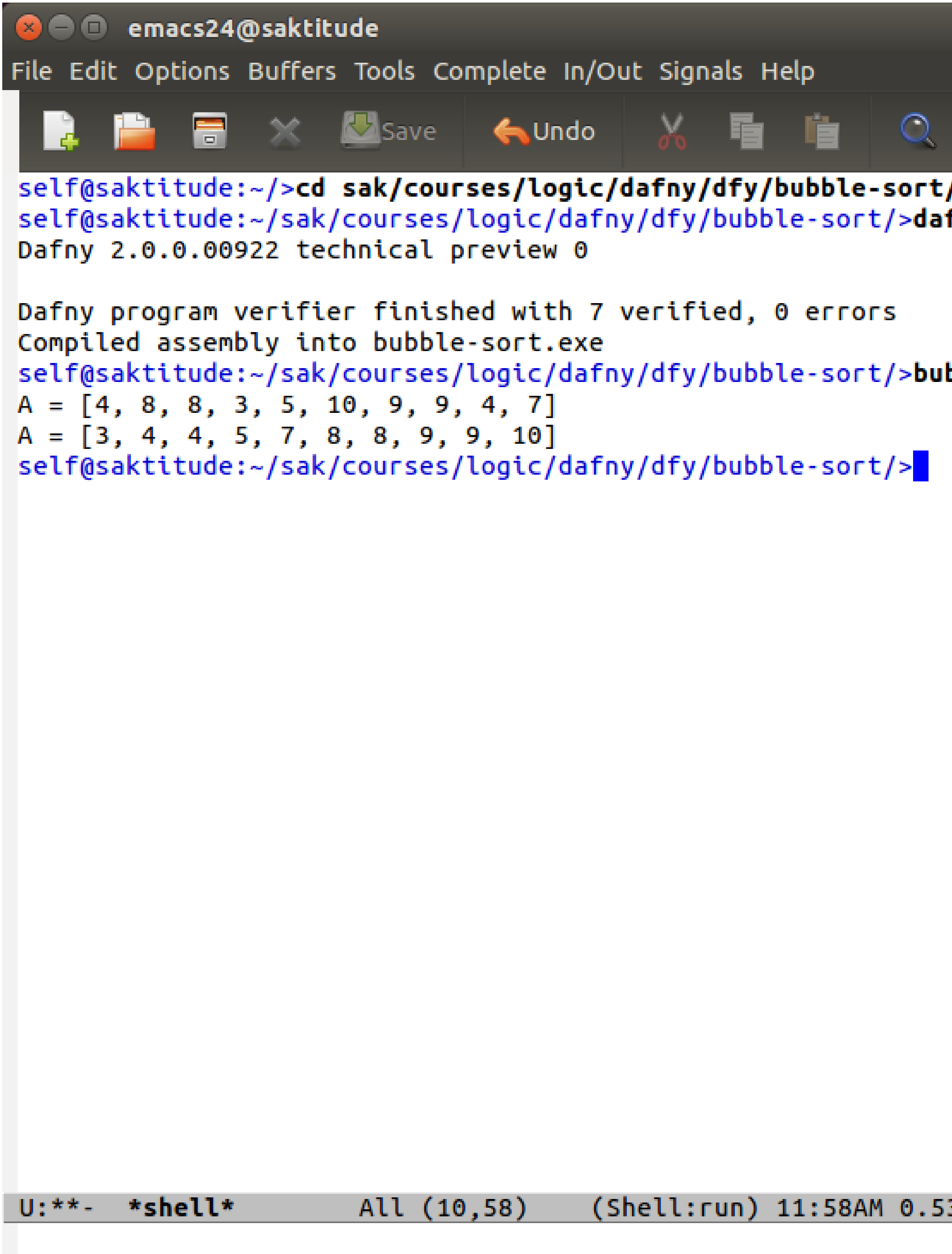

emacs24@saktitude
File Edit Options Buffers Tools YASnippet Hide/Show Help
Save
Undo
U:---  bubble-sort.dfy   46% (59,0)     (Dafny hs yas cor

```
      decreases A.Length - i
    {
      bubble (A, i);
      i ≔ i+1;
    }
  }
}

method bubble (A:array<int>, i:int)
  requires A ≠ null ∧ 0 ≤ i < A.Length
  requires partOrdered (A, 0, i)
  modifies A
  ensures partOrdered (A, 0, i+1)
  ensures permutation (A[..], old(A[..]))
{
  var j ≔ i;
  while j > 0 ∧ A[j-1] > A[j]
    invariant 0 ≤ j ≤ i
    invariant partOrdered (A, 0,j) ∧ partOrdered (A, j, i
    invariant permutation (A[..], old(A[..]))
    // Every element in A[0..j-1] <= every element in A[j
    invariant 1 < j+1 ≤ i ⟹ A[j-1] ≤ A[j+1]
    decreases j
  {
    A[j-1], A[j] ≔ A[j], A[j-1];
    j ≔ j-1;
  }
}


method Main ()
{
  var A ≔ new int[10];
  A[0],A[1],A[2],A[3],A[4],A[5],A[6],A[7],A[8],A[9] ≔
    //   10 9 8 7 6 5 4 3 2 1;
```



emacs24@saktitude
File Edit Options Buffers Tools YASnippet Hide/Show Help
Save
Undo
U:---  bubble-sort.dfy   Bot (72,0)   (Dafny hs yas co

```
    invariant 0 ≤ j ≤ i
    invariant partOrdered (A, 0,j) ∧ partOrdered (A, j, i
    invariant permutation (A[..], old(A[..]))
    // Every element in A[0..j-1] <= every element in A[
    invariant 1 < j+1 ≤ i ⟹ A[j-1] ≤ A[j+1]
    decreases j
  {
    A[j-1], A[j] ≔ A[j], A[j-1];
    j ≔ j-1;
  }
}


method Main ()
{
  var A ≔ new int[10];
  A[0],A[1],A[2],A[3],A[4],A[5],A[6],A[7],A[8],A[9] ≔
    //    10,9,8,7,6,5,4,3,2,1;
    4,8,8,3,5,10,9,9,4,7;
    print "A = ", A[..], "\n";
    bubbleSort (A);
    print "A = ", A[..], "\n";

}
```


emacs24@saktitude
File Edit Options Buffers Tools Complete In/Out Signals Help
Save
Undo
U:**-  *shell*  All (10,58)  (Shell:run) 11:58AM 0.5

```
self@saktitude:~/>cd sak/courses/logic/dafny/dfy/bubble-sort/
self@saktitude:~/sak/courses/logic/dafny/dfy/bubble-sort/>daf
Dafny 2.0.0.00922 technical preview 0

Dafny program verifier finished with 7 verified, 0 errors
Compiled assembly into bubble-sort.exe
self@saktitude:~/sak/courses/logic/dafny/dfy/bubble-sort/>bub
A = [4, 8, 8, 3, 5, 10, 9, 9, 4, 7]
A = [3, 4, 4, 5, 7, 8, 8, 9, 9, 10]
self@saktitude:~/sak/courses/logic/dafny/dfy/bubble-sort/>
```