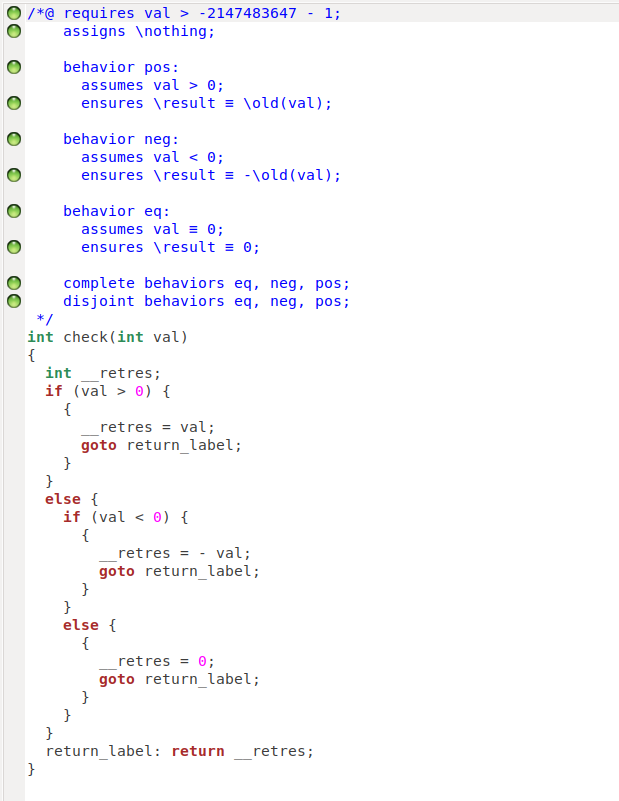
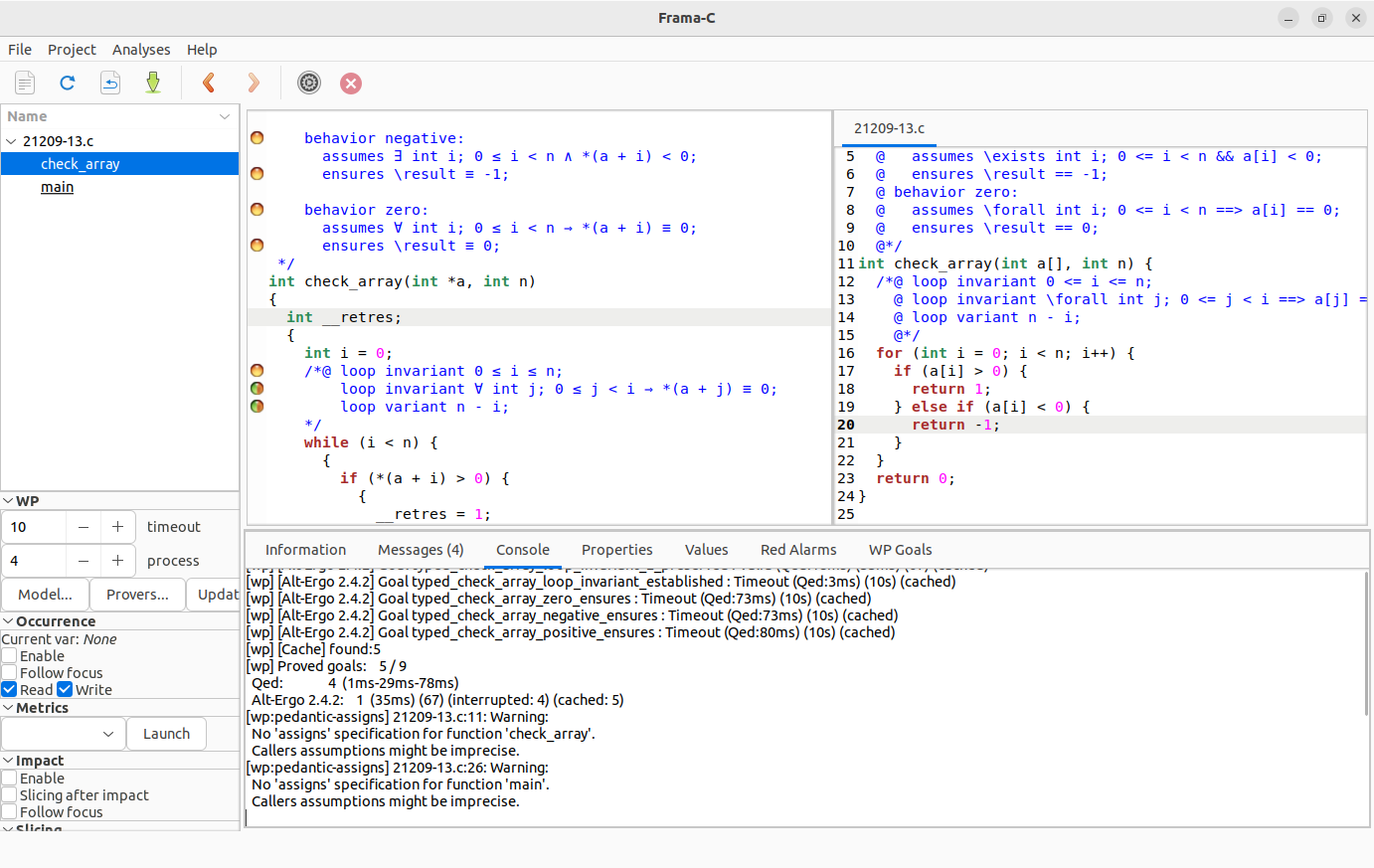
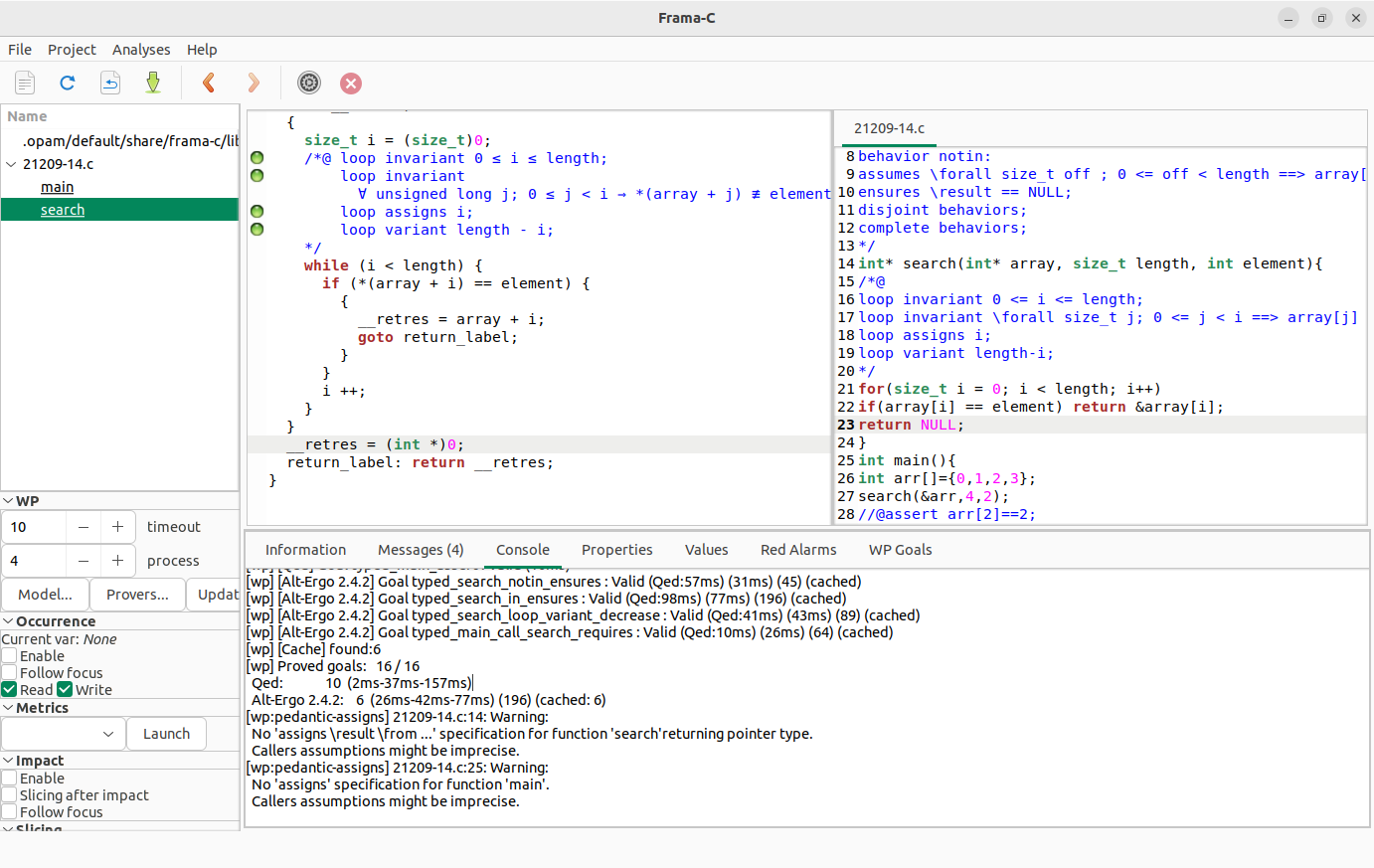
FRAMA C – ARRAYS

-BL.EN.U4CSE21171

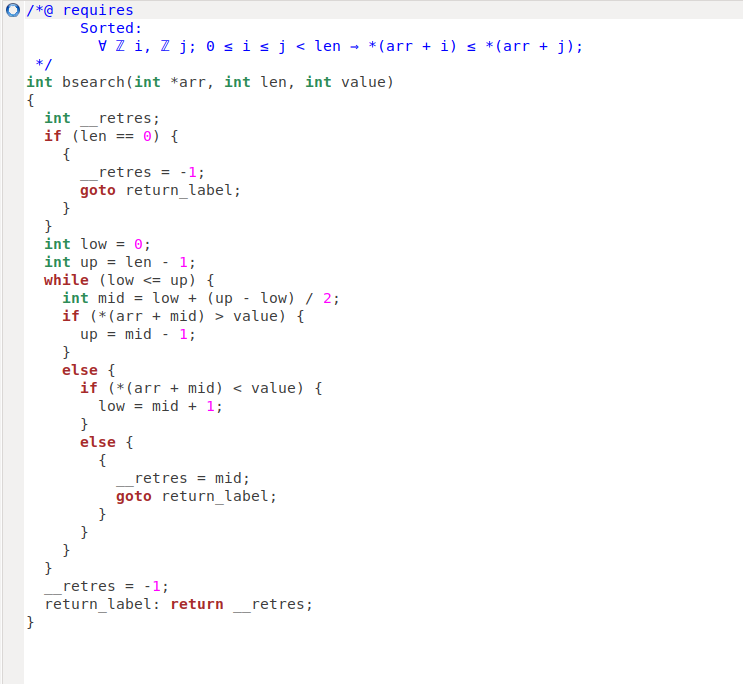
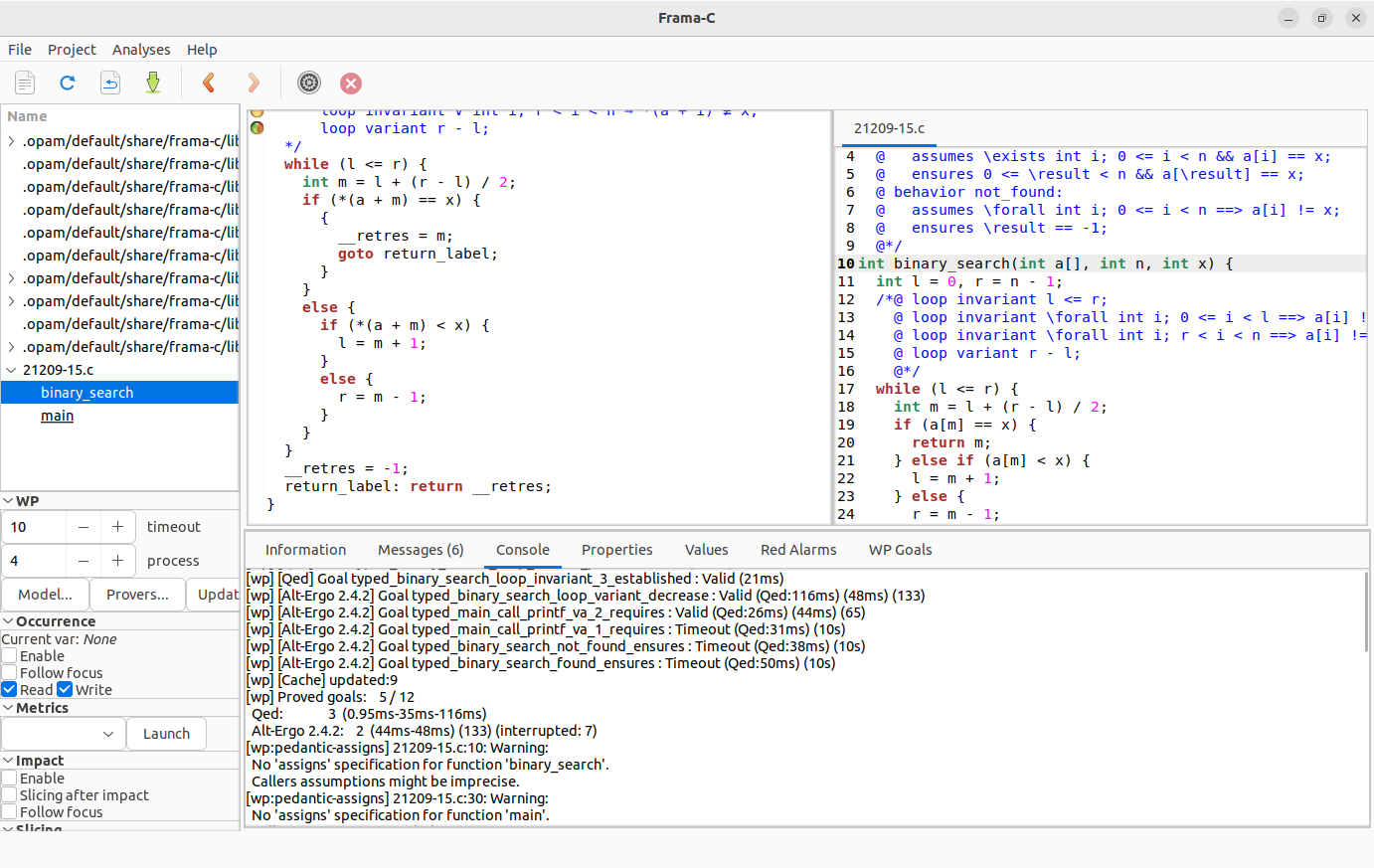
14. To check the positive or negative no. in an array using behavior



15. Linear Search problem using behavior



16. Binary Search problem using behavior



17. Bubble Sort algorithm using logical predicates/axioms/lemmas

/\*@

requires 0<n<INT\_MAX;

requires \valid(arr+(0..n-1));

ensures \forall integer i;

0<=i<n-1 ==> arr[n-1]>=arr[i];

\*/

void BubbleSort(int arr[],int n)

{

/\*@

loop invariant \forall integer x;0<=x<=i ==> arr[x]<=arr[i];

loop invariant 0<=i<=n-1;

loop assigns i,arr[0..n-1];

loop variant n-1-i;

\*/

for(int i=0;i<n-1;i++)

{

if(arr[i]>arr[i+1])

{

int temp = arr[i];

arr[i] = arr[i+1];

arr[i+1] = temp;

}

}

}

int main()

{

int len=5;

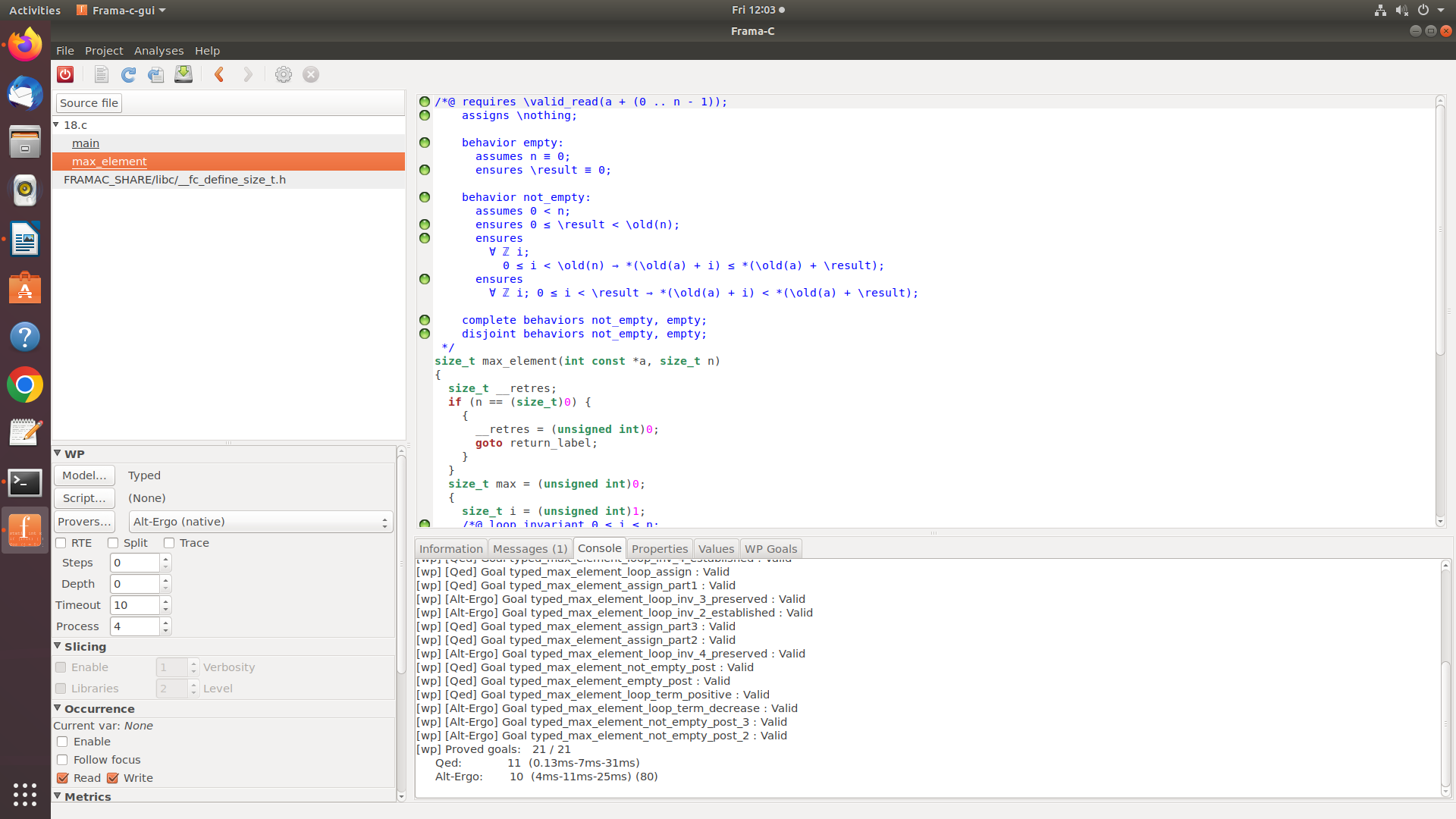
int a[]={4,6,2,88,22};

}

18. Insertion Sort algorithm using logical predicates/axioms/lemmas

/\*@  
requires 0<n<INT\_MAX;  
requires \valid(arr+(0..n-1));  
ensures \forall integer i,j;  
0<=j<i<n ==> arr[j]<=arr[i];  
\*/  
void SelectionSort(int arr[],int n)  
{  
int i,j,temp;  
/\*@  
loop invariant \forall integer p,q;  
0<=p<=q<n ==> arr[p]<=arr[q];  
loop invariant 0<=i<=n;  
loop assigns temp,i,j,arr[0..n-1];  
loop variant n-i;  
\*/  
for(i=0;i<n;i++)  
{  
/\*@  
loop invariant \forall integer p;  
i+1<=p<n ==> arr[p]<=arr[n];  
loop invariant i+1<=j<=n;  
loop assigns temp,j,arr[0..n-1];  
loop variant n-j;  
\*/  
for(j=i+1;j<n;j++)  
{  
if(arr[i]>arr[j])  
{  
temp = arr[i];  
arr[i] = arr[j];  
arr[j] = arr[i];  
}  
}  
}  
}  
  
int main()  
{  
int a=5  
int arr[]={4,5,6,8,3}  
  
}

19. Maximum and Minimum element in an array



20. Reverse an array

