Course: Programming Fundamental - ENSF 337

Lab #: Lab 5

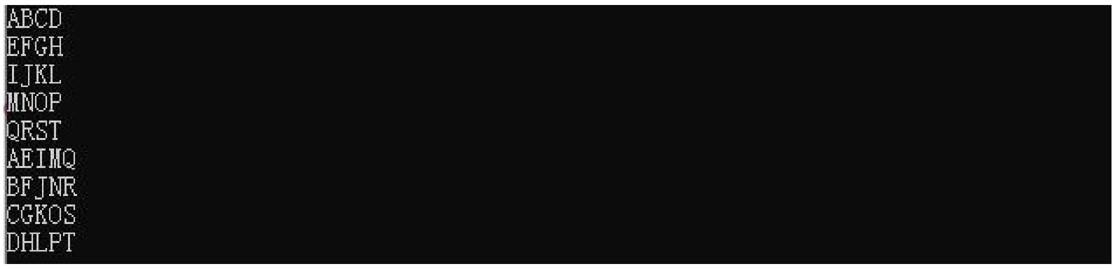
Instructor: Moussavi, Mahmood

Student Name: Shanzi Ye

Lab Section: B01

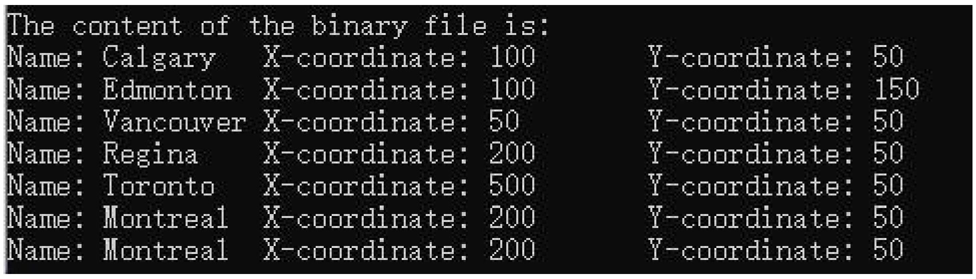
Date Submitted: June 15, 2022

Question A:



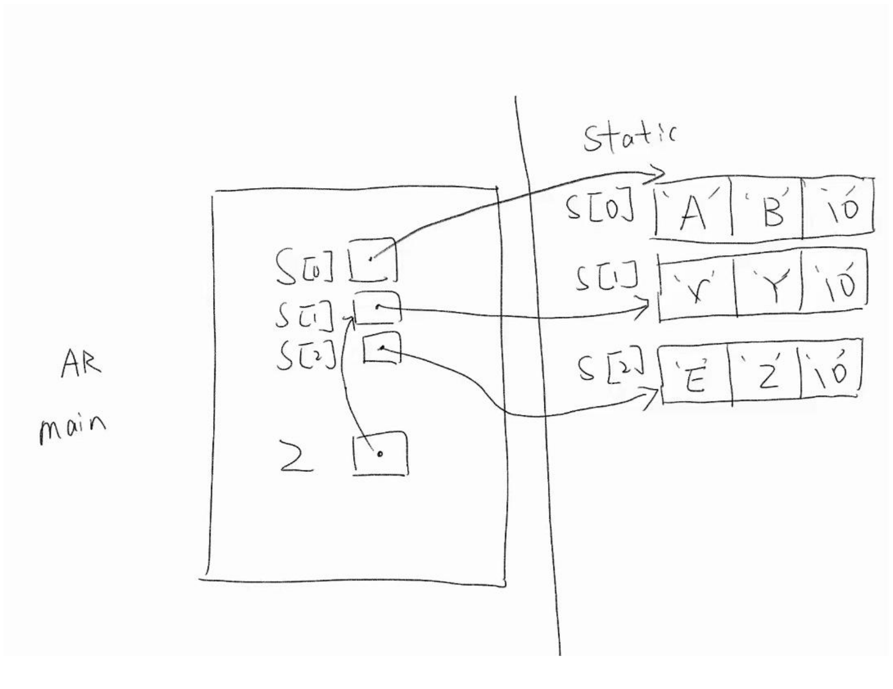
```
ABCD
EFGH
IJKL
MNOP
QRST
AEIMQ
BFJNR
CGKOS
DHLPT
```

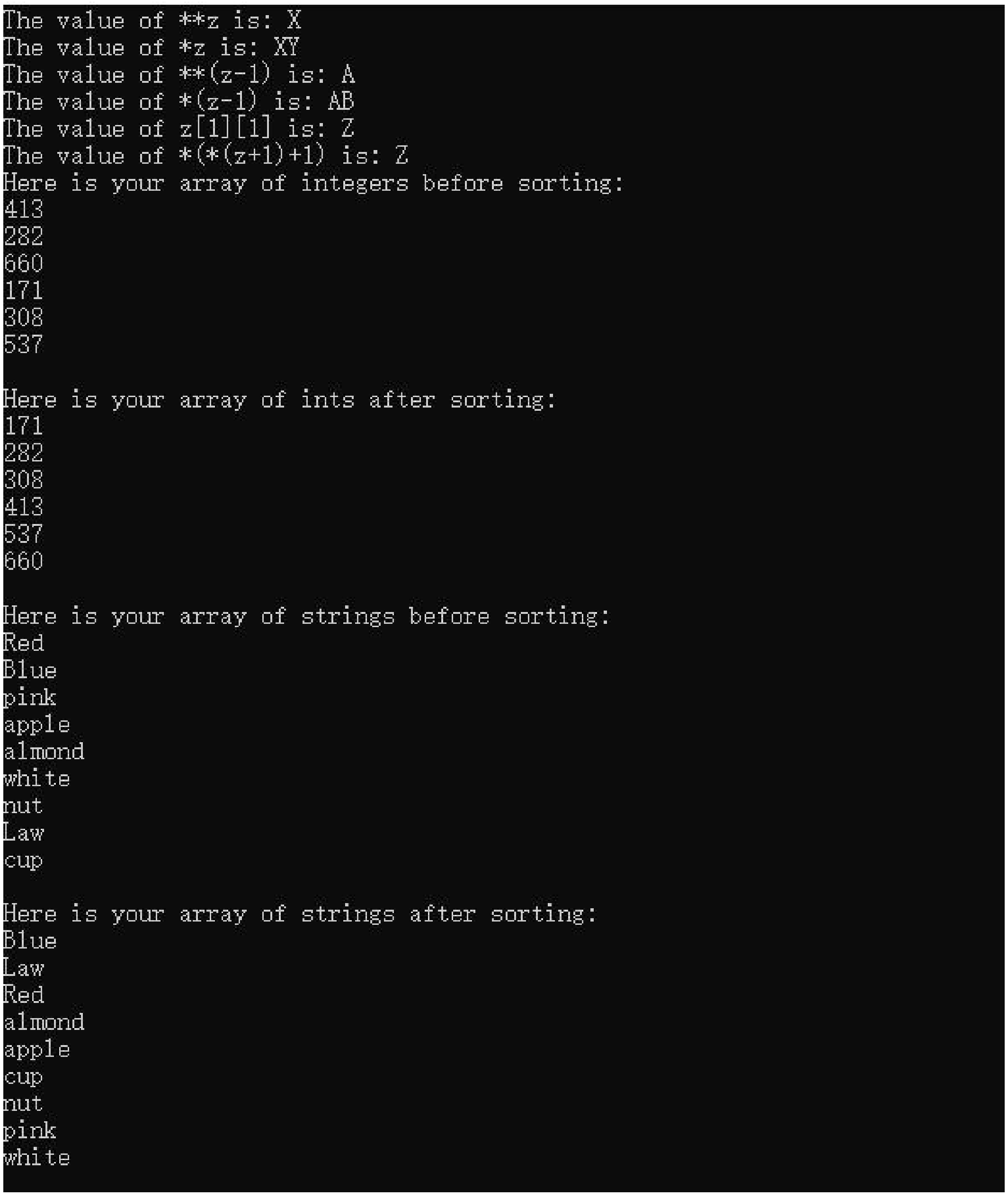
Question B:



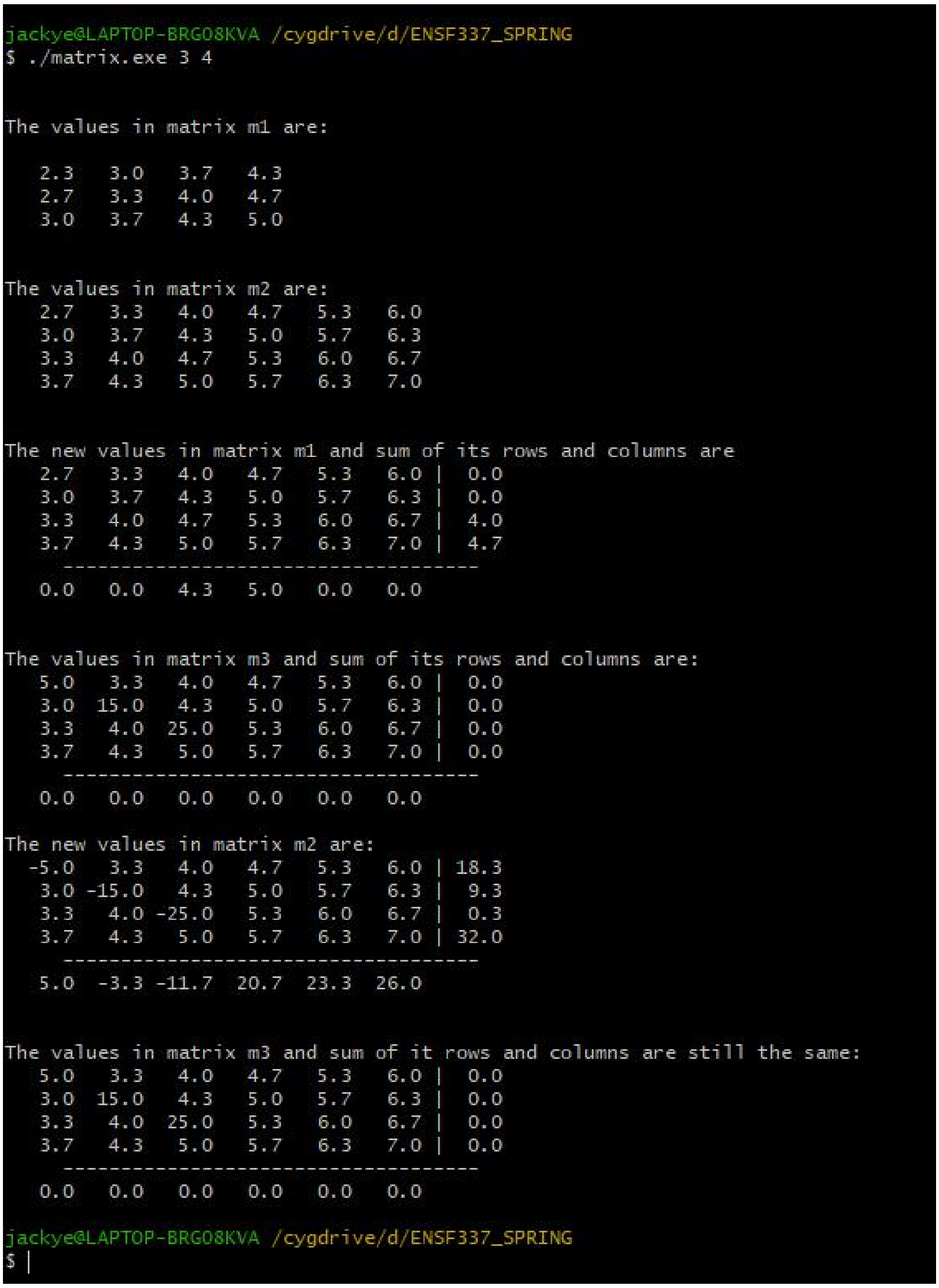
```
The content of the binary file is:
Name: Calgary   X-coordinate: 100       Y-coordinate: 50
Name: Edmonton  X-coordinate: 100       Y-coordinate: 150
Name: Vancouver X-coordinate: 50        Y-coordinate: 50
Name: Regina    X-coordinate: 200       Y-coordinate: 50
Name: Toronto   X-coordinate: 500       Y-coordinate: 50
Name: Montreal  X-coordinate: 200       Y-coordinate: 50
Name: Montreal  X-coordinate: 200       Y-coordinate: 50
```

Question C:



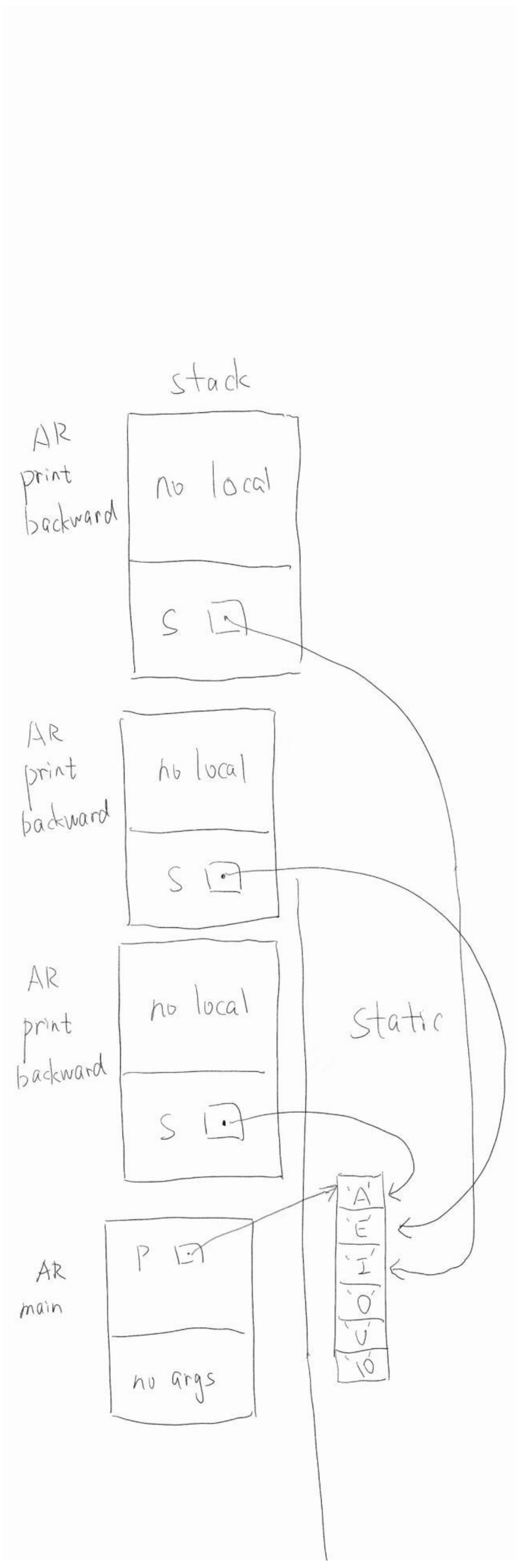

```
The value of **z is: X
The value of *z is: XY
The value of **(z-1) is: A
The value of *(z-1) is: AB
The value of z[1][1] is: Z
The value of *(*(z+1)+1) is: Z
Here is your array of integers before sorting:
413
282
660
171
308
537

Here is your array of ints after sorting:
171
282
308
413
537
660

Here is your array of strings before sorting:
Red
Blue
pink
apple
almond
white
nut
Law
cup

Here is your array of strings after sorting:
Blue
Law
Red
almond
apple
cup
nut
pink
white
```

Question D:

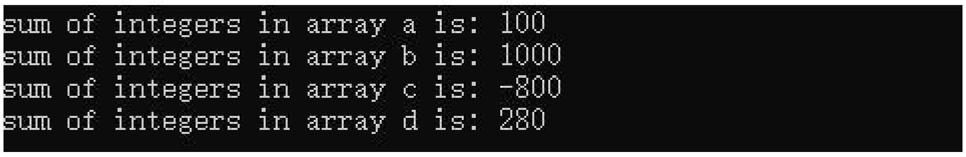


```
jackye@LAPTOP-BRG08KVA /cygdrive/d/ENSF337_SPRING
$ ./matrix.exe 3 4


The values in matrix m1 are:

   2.3   3.0   3.7   4.3
   2.7   3.3   4.0   4.7
   3.0   3.7   4.3   5.0


The values in matrix m2 are:
   2.7   3.3   4.0   4.7   5.3   6.0
   3.0   3.7   4.3   5.0   5.7   6.3
   3.3   4.0   4.7   5.3   6.0   6.7
   3.7   4.3   5.0   5.7   6.3   7.0


The new values in matrix m1 and sum of its rows and columns are
   2.7   3.3   4.0   4.7   5.3   6.0 |  0.0
   3.0   3.7   4.3   5.0   5.7   6.3 |  0.0
   3.3   4.0   4.7   5.3   6.0   6.7 |  4.0
   3.7   4.3   5.0   5.7   6.3   7.0 |  4.7
     ------------------------------------
   0.0   0.0   4.3   5.0   0.0   0.0


The values in matrix m3 and sum of its rows and columns are:
   5.0   3.3   4.0   4.7   5.3   6.0 |  0.0
   3.0  15.0   4.3   5.0   5.7   6.3 |  0.0
   3.3   4.0  25.0   5.3   6.0   6.7 |  0.0
   3.7   4.3   5.0   5.7   6.3   7.0 |  0.0
     ------------------------------------
   0.0   0.0   0.0   0.0   0.0   0.0

The new values in matrix m2 are:
  -5.0   3.3   4.0   4.7   5.3   6.0 | 18.3
   3.0 -15.0   4.3   5.0   5.7   6.3 |  9.3
   3.3   4.0 -25.0   5.3   6.0   6.7 |  0.3
   3.7   4.3   5.0   5.7   6.3   7.0 | 32.0
     ------------------------------------
   5.0  -3.3 -11.7  20.7  23.3  26.0


The values in matrix m3 and sum of it rows and columns are still the same:
   5.0   3.3   4.0   4.7   5.3   6.0 |  0.0
   3.0  15.0   4.3   5.0   5.7   6.3 |  0.0
   3.3   4.0  25.0   5.3   6.0   6.7 |  0.0
   3.7   4.3   5.0   5.7   6.3   7.0 |  0.0
     ------------------------------------
   0.0   0.0   0.0   0.0   0.0   0.0

jackye@LAPTOP-BRG08KVA /cygdrive/d/ENSF337_SPRING
$ |
```

Question E:

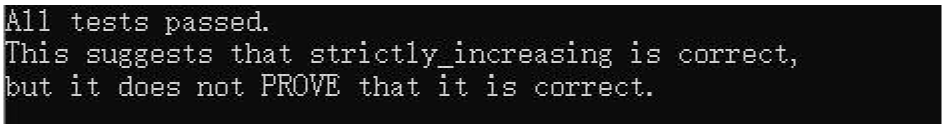

stack
AR print backward
no local
S
AR print backward
no local
S
AR print backward
no local
S
static
AR main
P
no args
'A'
'E'
'I'
'O'
'U'
'\0'

Question F:



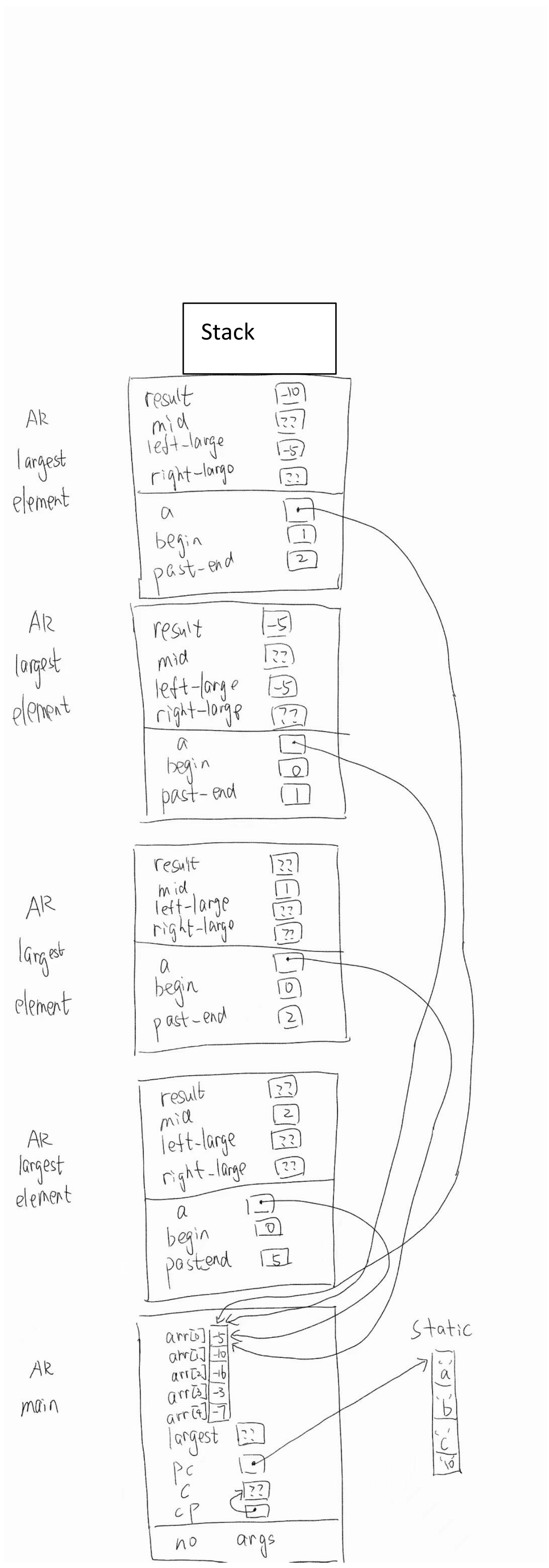
```
sum of integers in array a is: 100
sum of integers in array b is: 1000
sum of integers in array c is: -800
sum of integers in array d is: 280
```

Question G:



```
All tests passed.
This suggests that strictly_increasing is correct,
but it does not PROVE that it is correct.
```

Question H:


Stack
AR largest element
result -10
mid ??
left-large -5
right-large ??
a
begin 1
past-end 2
AR largest element
result -5
mid ??
left-large -5
right-large ??
a
begin 0
past-end 1
AR largest element
result ??
mid 1
left-large ??
right-large ??
a
begin 0
past-end 2
AR largest element
result ??
mid 2
left-large ??
right-large ??
a
begin 0
pastend 5
AR main
arr[0] -5
arr[1] -10
arr[2] -16
arr[3] -3
arr[4] -7
largest ??
pc
c ??
cp
no args
Static
'a'
'b'
'c'
'\0'

Question I:

```
/*
* File Name: OLList.cpp
* Assignment: Lab5 Exercise I
* Lab section: (B01)
* Completed by: Shanzi Ye
* Submission Date: June 14, 2022
*/

#include <iostream>
using namespace std;
#include <stdlib.h>
#include "OLList2.h"

OLList::OLList()
    : headM(0)
{
}

OLList::OLList(const OLList& source)
{
    copy(source);
}

OLList::~OLList()
{
    destroy();
}

OLList& OLList::operator =(const OLList& rhs)
{
    if (this != &rhs) {
        destroy();
        copy(rhs);
    }
    return *this;
}

void OLList::insert(const ListItem& itemA)
{
    Node *new_node = new Node;
    new_node->item = itemA;

    if (headM == 0 || itemA <= headM->item) {
        new_node->next = headM;
        headM = new_node;
    }
    else {
        Node *before = headM;            // will point to node in front of new node
        Node *after = headM->next; // will be 0 or point to node after new node
        while(after != 0 && itemA > after->item) {
            before = after;
            after = after->next;
        }
        new_node->next = after;
        before->next = new_node;
    }
}

void OLList::remove(const ListItem& itemA)
{
    if (headM == 0 || itemA < headM->item)
        return;

    Node *doomed_node = 0;
    if (itemA == headM->item) {
        doomed_node = headM;
        headM = headM->next;
    }
    else {
        Node *before = headM;
        Node *maybe_doomed = headM->next;
        while(maybe_doomed != 0 && itemA > maybe_doomed->item) {
            before = maybe_doomed;
            maybe_doomed = maybe_doomed->next;
        }
        if (maybe_doomed != 0 && maybe_doomed->item == itemA) {
            doomed_node = maybe_doomed;
            before->next = maybe_doomed->next;
        }
    }
    delete doomed_node;                    // Does nothing if doomed_node == 0.
}

void OLList::print() const
```

```
{
   if (headM == 0)
      cout << "    LIST IS EMPTY.\n";
   else
      for (Node *p = headM; p != 0; p = p->next)
         cout << "    " << p->item << '\n';
}

void OLList::copy(const OLList& source)
{
   // The next line doesn't do anything.    It justs shuts up the compiler
   // warning about an unused argument.
   (void) source;

   // Print an error message and terminate the program.
   cout << "\nOLList::copy is not implemented properly, so the program"
          << " is calling exit.\n";
   exit(1);

}

void OLList::destroy()
{
   destroy_sublist(headM);
}

void OLList::destroy_sublist(Node *sublist_head)
{
   if (sublist_head != 0) {
      destroy_sublist(sublist_head->next);



      // point one

      delete sublist_head;
   }
}
Node* OLList::copy_sublist(const Node* source_sublist)
{
      const Node* current = source_sublist;
      if (current == NULL) return NULL;
      else {
            Node* newNode = new Node;
```

```
        newNode->item = current->item;
        newNode->next = copy_sublist(current->next);
        return(newNode);
    }
}
```