Course: Programming Fundamental – ENSF 337

Lab #: Lab 3  
Instructor: Mansouri Habibabadi  
Student Name: Drew Hengehold

Lab Section: B01

Date submitted: October 5, 2022

UCID: 30151823

Lab1\_Excersize A





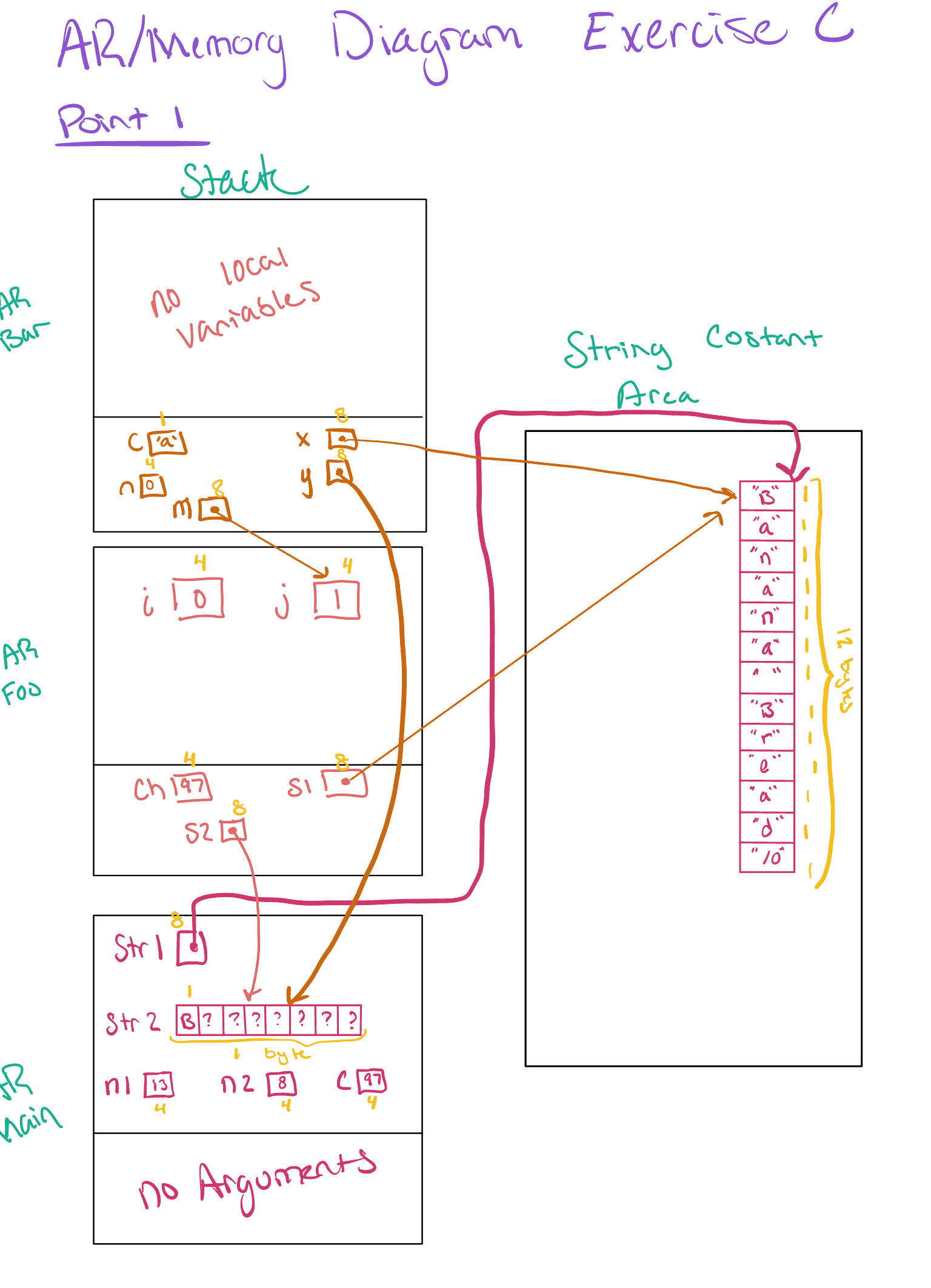
Lab1\_Excersize B

**AR/Memory Diagram for Exercise B**

Diagram

Description automatically generated

Lab1\_Excersize C



Diagram

Description automatically generated

Lab1\_Excersize D

**Source Code:**

/\*

\* lab3exe\_D.c

\* ENSF 337, lab3 Exercise D

\* Author of pascal\_triangle function = Drew Hengehold

\* In this program the implementatiom of function pascal\_trangle is missing.

\* Studtent must complete this function.

\*/

#include <stdio.h>

#include <stdlib.h>

void pascal\_triangle(int n);

/\* REQUIRES: n > 0 and n <= 20

PROMISES: displays a pascal\_triangle. the first 5 line of the function's output

should have the following format:

row 0: 1

row 1: 1 1

row 2: 1 2 1

row 3: 1 3 3 1

row 4: 1 4 6 4 1

\*/

int main() {

int nrow;

// These are ALL of the variables you need!

printf("Enter the number of rows (Max 20): ");

scanf("%d", &nrow);

if(nrow <= 0 || nrow > 20) {

printf("Error: the maximum number of rows can be 20.\n");

exit(1);

}

pascal\_triangle(nrow);

return 0;

}

void pascal\_triangle(int n) {

int row[n];

int previous\_row[n];

int \*old = previous\_row;

int \*new = row;

for (int i = 0; i <= n; i++){

printf("Row %d:\t", i);

for(int j = 0; j <= i; j++){

if (j==0 || j==i)

new[j] = 1;

else{

new[j] = old[j-1] + old[j];

}

printf("%d\t", new[j]);

}

int \*origional;

origional = new;

new = old;

old = origional;

printf("\n");

}

}

Text

Description automatically generated

**Output with 9 Rows**

Lab1\_Excersize E

**Source Code**

/\* lab3exe\_E.c

\* ENSF 337, Lab 3 Exercise E

\* Author of Functions:Drew Hengehold

\*/

#include <stdio.h>

#include <string.h>

int substring(const char \*s1, const char \*s2);

/\* REQUIRES

\* s1 and s2 are valid C-string terminated with '\0';

\* PROMISES

\* returns one if s2 is a substring of s1). Otherwise returns zero.

\*/

void select\_negatives(const int \*source, int n\_source,

int\* negatives\_only, int\* number\_of\_negatives);

/\* REQUIRES

\* n\_source >= 0.

\* Elements source[0], source[1], ..., source[n\_source - 1] exist.

\* Elements negatives\_only[0], negatives\_only[1], ..., negatives\_only[n\_source - 1] exist.

\* PROMISES

\* number\_of\_negatives == number of negative values in source[0], ..., source[n\_source - 1].

\* negatives\_only[0], ..., negatives\_only[number\_of\_negatives - 1] contain those negative values, in

\* the same order as in the source array. \*/

int main(void)

{

char s[] = "Knock knock! Who's there?";

int a[] = { -10, 9, -17, 0, -15 };

int size\_a;

int i;

int negative[5];

int n\_negative;

size\_a = sizeof(a) / sizeof(a[0]);

printf("a has %d elements:", size\_a);

for (i = 0; i < size\_a; i++)

printf(" %d", a[i]);

printf("\n");

select\_negatives(a, size\_a, negative, &n\_negative);

printf("\nnegative elements from array a are as follows:\n");

for (i = 0; i < n\_negative; i++){

printf(" %d", negative[i]);

printf("\n");

}

printf("\nNow testing substring function....\n");

printf("Answer must be 1. substring function returned: %d\n" , substring(s, "Who"));

printf("Answer must be 0. substring function returned: %d\n" , substring(s, "knowk"));

printf("Answer must be 1. substring function returned: %d\n" , substring(s, "knock"));

printf("Answer must be 0. substring function returned: %d\n" , substring(s, ""));

printf("Answer must be 1. substring function returned: %d\n" , substring(s, "ck! Who's"));

printf("Answer must be 0. substring function returned: %d\n" , substring(s, "ck!Who's"));

return 0;

}

int substring(const char \*s1, const char\* s2)

{

int count1 = 0;

int count2 = 0;

while(s1[count1] != '\0')

count1++;

while(s2[count2] != '\0')

count2++;

int counter = 0;

for(int i = 0; i < count1; i++){

if(s1[i] == s2[counter]){

counter++;

}

else

{

counter = 0;

}

if(counter == count2){

return 1;

break;

}

}

return 0;

}

void select\_negatives(const int \*source, int n\_source,

int\* negatives\_only, int\* number\_of\_negatives)

{

int counter = 0;

for(int i = 0; i <= n\_source; i++){

if(source[i] < 0){

negatives\_only[counter] = source[i];

counter++;

}

}

\*number\_of\_negatives = counter;

return;

}

Text

Description automatically generated

**Ran with my own data set // Data set below:**

Array s[] was “HelloWorld”, the sub strings were “Wo”, “Hello”, “W”, “gro”, “ld”, and “Hellor” respectively.

**Text

Description automatically generated**

**Ran with the sample data set given**

Lab1\_Excersize F

**Source Code:**

/\* File: palindrome.c

\* ENSF 337

\* Exercise F - Lab 3

\* Abstract: The program receives a string (one or more words) and indicates

\* if the string is a palindrome or not. Plaindrome is a phrase that spells the

\* same from both ends

\* Author of Functions: Drew Hengehold

\*/

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define SIZE 100

/\* function prototypes\*/

int is\_palindrome (const char \*str);

/\* REQUIRES: str is pointer to a valid C string.

\* PROMISES: the return value is 1 if the string a is palindrome.\*/

void strip\_out(char \*str);

/\* REQUIRES: str points to a valid C string terminated with '\0'.

\* PROMISES: strips out any non-alphanumerical characters in str\*/

int main(void)

{

int p =0;

char str[SIZE], temp[SIZE];

fgets(str, SIZE, stdin);

/\* Remove end-of-line character if there is one in str.\*/

if (str[strlen(str) - 1] == '\n')

str[strlen(str) - 1] = '\0';

strcpy(temp,str);

/\* This loop is infinite if the string "done" never appears in the

\* input. That's a bit dangerous, but OK in a test harness where

\* the programmer is controlling the input. \*/

while(strcmp(str, "done") !=0) /\* Keep looping unless str matches "done". \*/

{

#if 1

strip\_out(str);

p = is\_palindrome(str);

#endif

if(!p)

printf("\n \"%s\" is not a palindrome.", temp);

else

printf("\n \"%s\" is a palindrome.", temp);

fgets(str, SIZE, stdin);

/\* Remove end-of-line character if there is one in str.\*/

if(str[strlen(str) - 1] == '\n')

str[strlen(str) - 1]= '\0';

strcpy(temp, str);

}

return 0;

}

void strip\_out(char \*str){

char str\_two[strlen(str)];

int str\_two\_counter = 0;

for(int i = 0; i<(strlen(str)); i++){

char element = str[i];

if(isalnum(element))

{

str\_two[str\_two\_counter] = (char)element;

str\_two\_counter++;

}

}

str\_two[str\_two\_counter] = '\0';

strcpy(str, str\_two);

}

int is\_palindrome (const char \*str){

int counter = 0;

for(int i = 0, j = (int)(strlen(str)-1); i<(strlen(str)-1)|| j>=0; i++, j--){

if(tolower(str[i]) == tolower(str[j]))

counter++;

}

if(counter == (strlen(str))){

return 1;

}

return 0;

}

Text

Description automatically generated

**Screenshots of Palindrome output**