Course: ENSF 337 - Programming Fundamentals for Software and Computer

Lab #: 3

Instructor: Dr. Maan Khedr

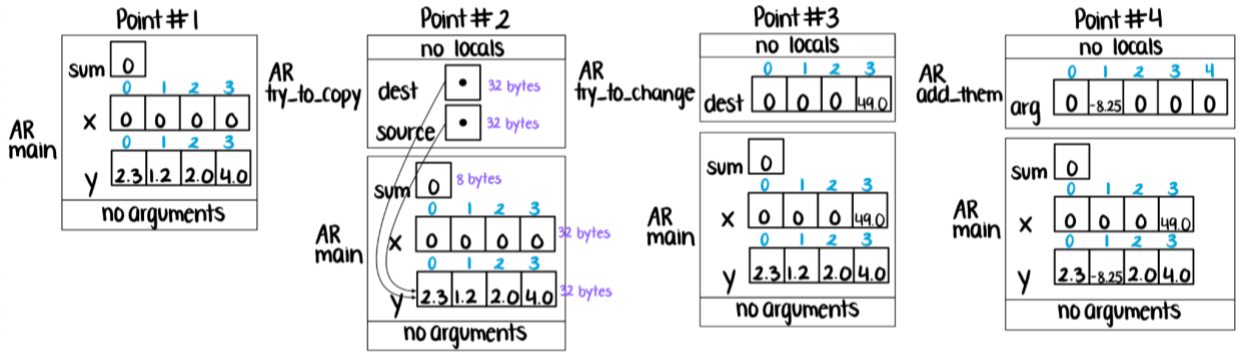
Student Name: Sadia Khandaker

Lab Section: B04

Date Submitted: October 14, 2021

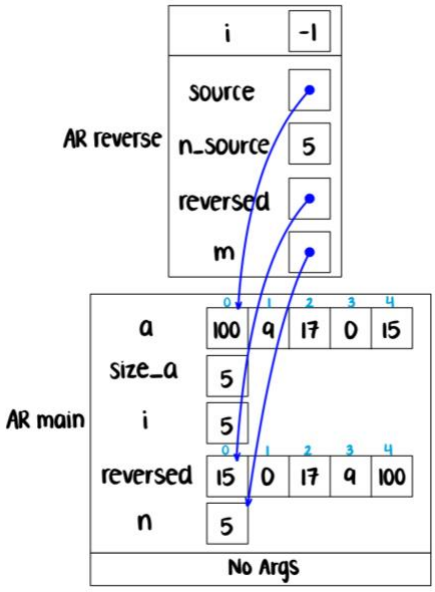
Exercise A: Built-in Arrays in C

AR Diagrams at Point 1, 2, 3, and 4:



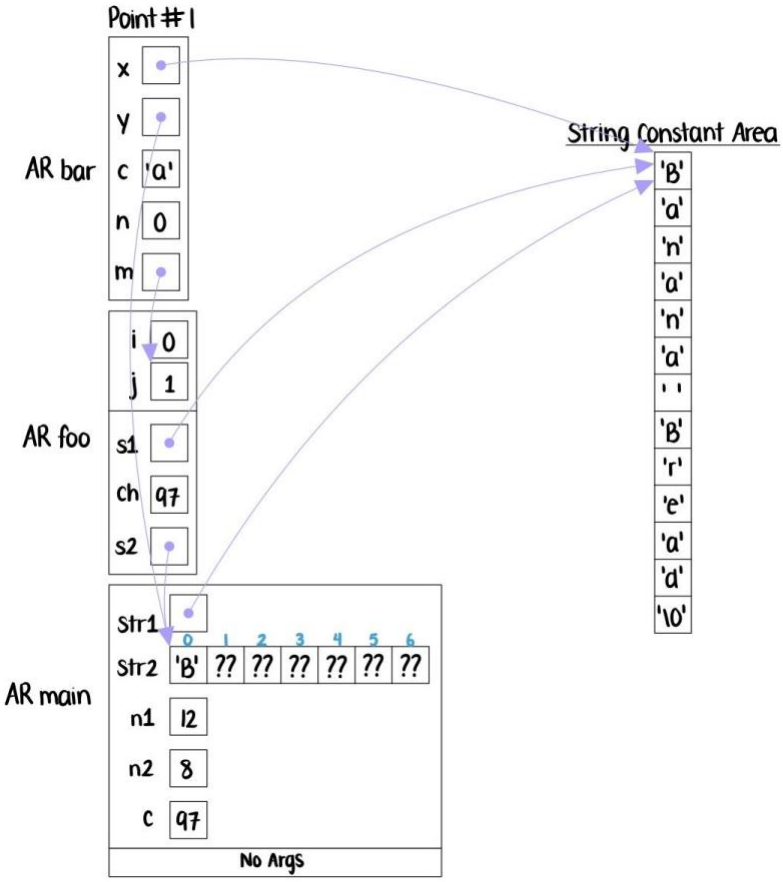
Exercise B: AR Diagrams With Arrays

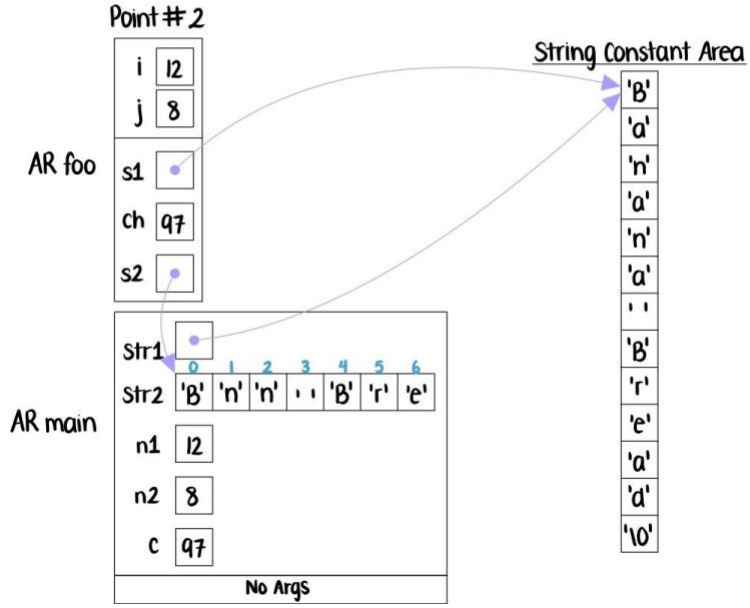
AR Diagram at Point 1:



Exercise C: AR Diagrams With C-string

AR Diagram at Point 1, and Point 2:





Exercise D: Problem Solving

Pascal's Triangle Program:

```c
/*      File Name: lab3exe_D.c
        *  Assignment: Lab 3 Exercise D
        *  Lab section: B04
        *  Completed by: Sadia Khandaker
        *  Submission Date: October 14, 2021
        */
#include <stdio.h>
#include <stdlib.h>
void pascal_triangle(int n);

int main() {
    int nrow;
    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 i, j, c = 1;
    for (i = 0; i < n; i++) {
        printf("Row %d: ",i);
        for (j = 0; j <= i; j++) {
            if (i == 0 || j == 0) {
                c = 1;
            } else {
                c = c * (i - j + 1) / (j);
            }
            printf("%4d", c);
        }
        printf("\n");
    }
}
```

Output:

```
Enter the number of rows (Max 20): 9
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
Row 5:    1   5  10  10   5   1
Row 6:    1   6  15  20  15   6   1
Row 7:    1   7  21  35  35  21   7   1
Row 8:    1   8  28  56  70  56  28   8   1
```

Exercise E: Writing Functions That Work With Arrays

Manipulating C-string Program:

```
* File Name: lab3exe_E.c
 * Assignment: Lab 3 Exercise E
 * Lab Section: B04
 * Completed By: Sadia Khandaker
 * Submission Date: October 14, 2021
 */

#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:");
    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 i,j,k;
    for(i=0;s1[i] !='\0';i++){
        j=i;
        k=0;
        while(s2[k]!='\0'&& s1[j]==s2[k]){
            ++j;
            ++k;
        }
        if(s2[k]=='\0'&& k>0)
            return 1;
    }
    return 0;
}

void select_negatives(const int *source, int n_source,
                      int* negatives_only, int* number_of_negatives)
{
    int i, j=0;
    *number_of_negatives = 0;
    for(i = 0; i < n_source; i++) {
        if(source[i]<0) {
            negatives_only[j] = source[i];
            j++;
            *number_of_negatives = j;
        }
    }
}
```

Output:

```
a has 5 elements:  -10  9  -17  0  -15

negative elements from array a are as follows:  -10  -17  -15

Now testing substring function....
Answer must be 1. substring function returned: 1
Answer must be 0. substring function returned: 0
Answer must be 1. substring function returned: 1
Answer must be 0. substring function returned: 0
Answer must be 1. substring function returned: 1
Answer must be 0. substring function returned: 0
```

Exercise F: More Practice With Strings

Palindrome Program:

```
/*
 * File Name: lab3exe_F.c
 * Assignment: Lab 3 Exercise F
 * Lab Section: B04
 * Completed By: Sadia Khandaker
 * Submission Date: October 14, 2021
 */

#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 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 s;
    int i=0,j=0;
    while ((s = str[i++]) != '\0') {
        if (isalnum(s)) {
            str[j++] = s;
        }
    }
    str[j] = '\0';

}

int is_palindrome(const char *str) {
    int l = strlen(str);
    for (int i=0; i < l; ++i) {
        if (tolower(str[i]) != tolower(str[l-1-i])) {
            return 0;
        }
    }
    return 1;
}
```

Output:

```
"Radar" is a palindrome.
"Madam I'm Adam" is a palindrome.
"Alfalfa" is not a palindrome.
"He maps spam, eh?" is a palindrome.
"I did, did I?" is a palindrome.
"       I prefer pi." is a palindrome.
"Ed is on no side" is a palindrome.
"Am I loco, Lima?" is a palindrome.
"         Bar crab." is a palindrome.
"A war at Tarawa." is a palindrome.
"Ah, Satan sees Natasha" is a palindrome.
"     Borrow or rob?" is a palindrome.
"233332" is a palindrome.
"324556" is not a palindrome.
"Hello world!!" is not a palindrome.
"     Avon sees nova  " is a palindrome.
"Can I attain a 'C'?" is a palindrome.
"Sept 29, 2005." is not a palindrome.
"Delia failed." is a palindrome.
"Draw nine men $$  inward" is a palindrome.
```