$Course : Programming Fundamental – ENSF 337

Lab # : Lab 3

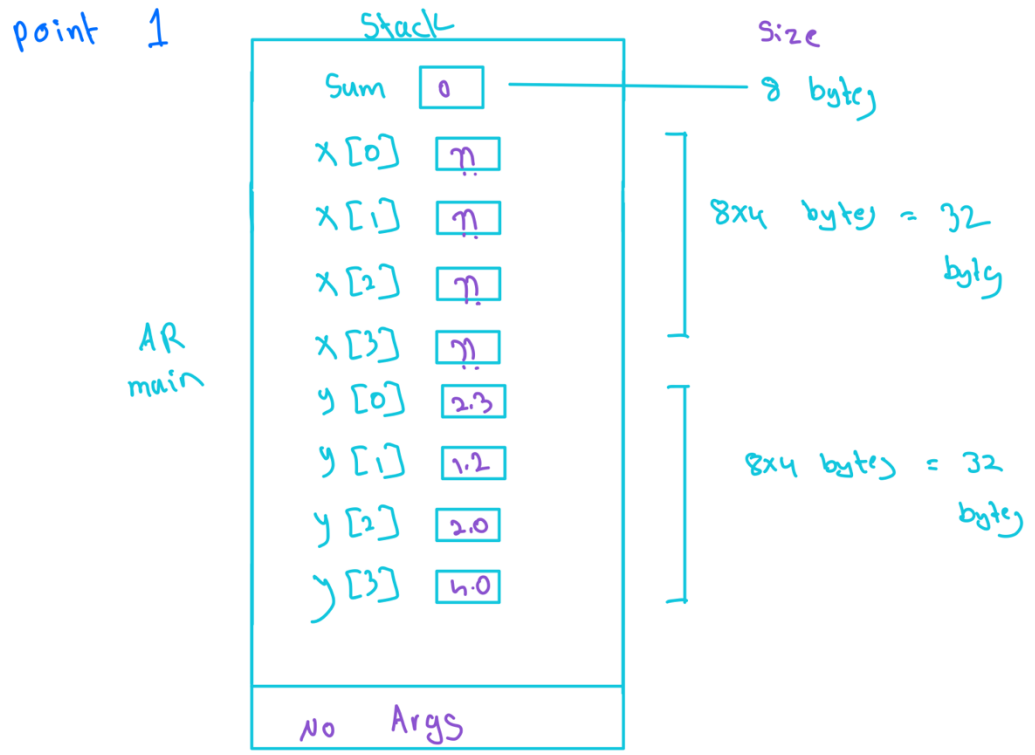
Instructor : M. Moussavi

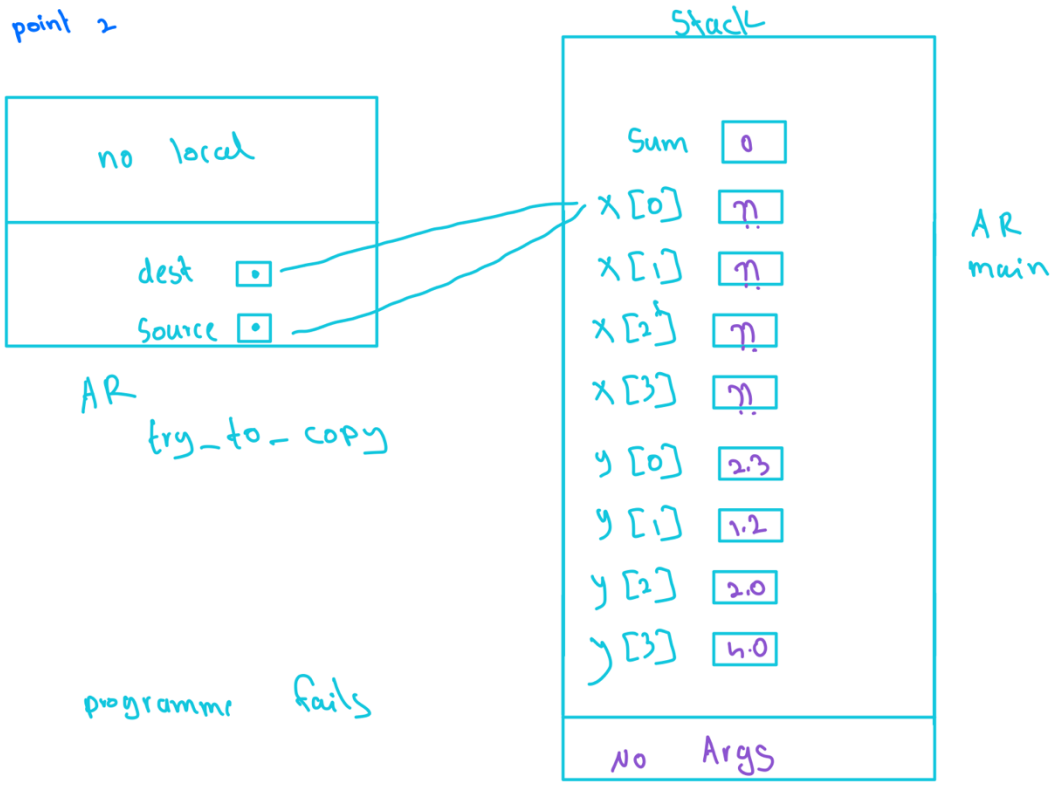
Student Name : Nimna Wijedasa

Lab Section : B02

Date submitted : Oct 6 , 2022

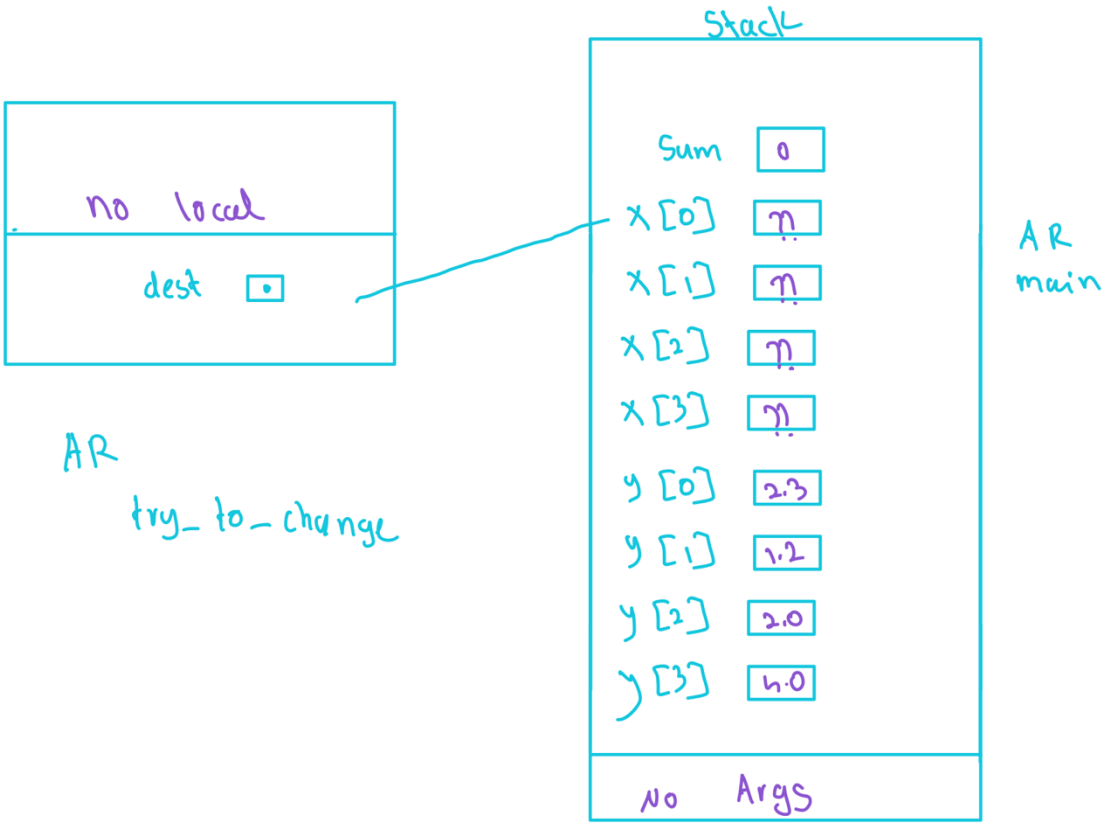
Exercise A



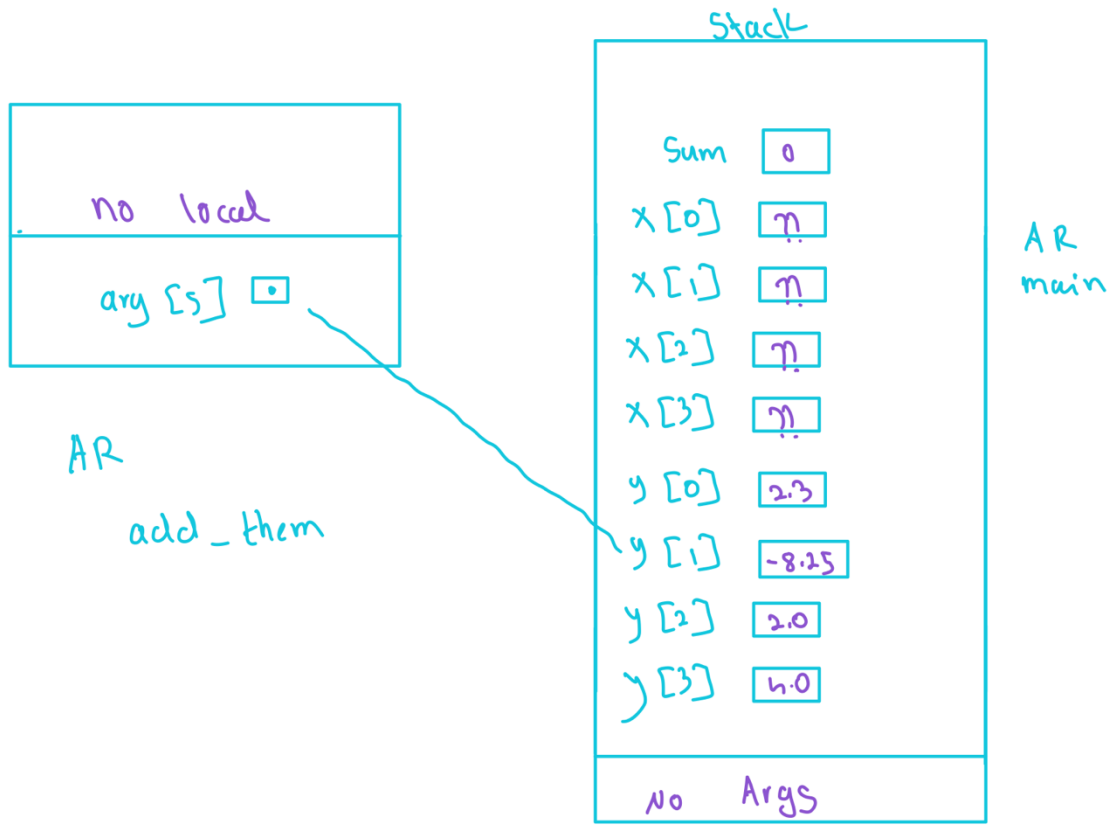


programme fails

point 3

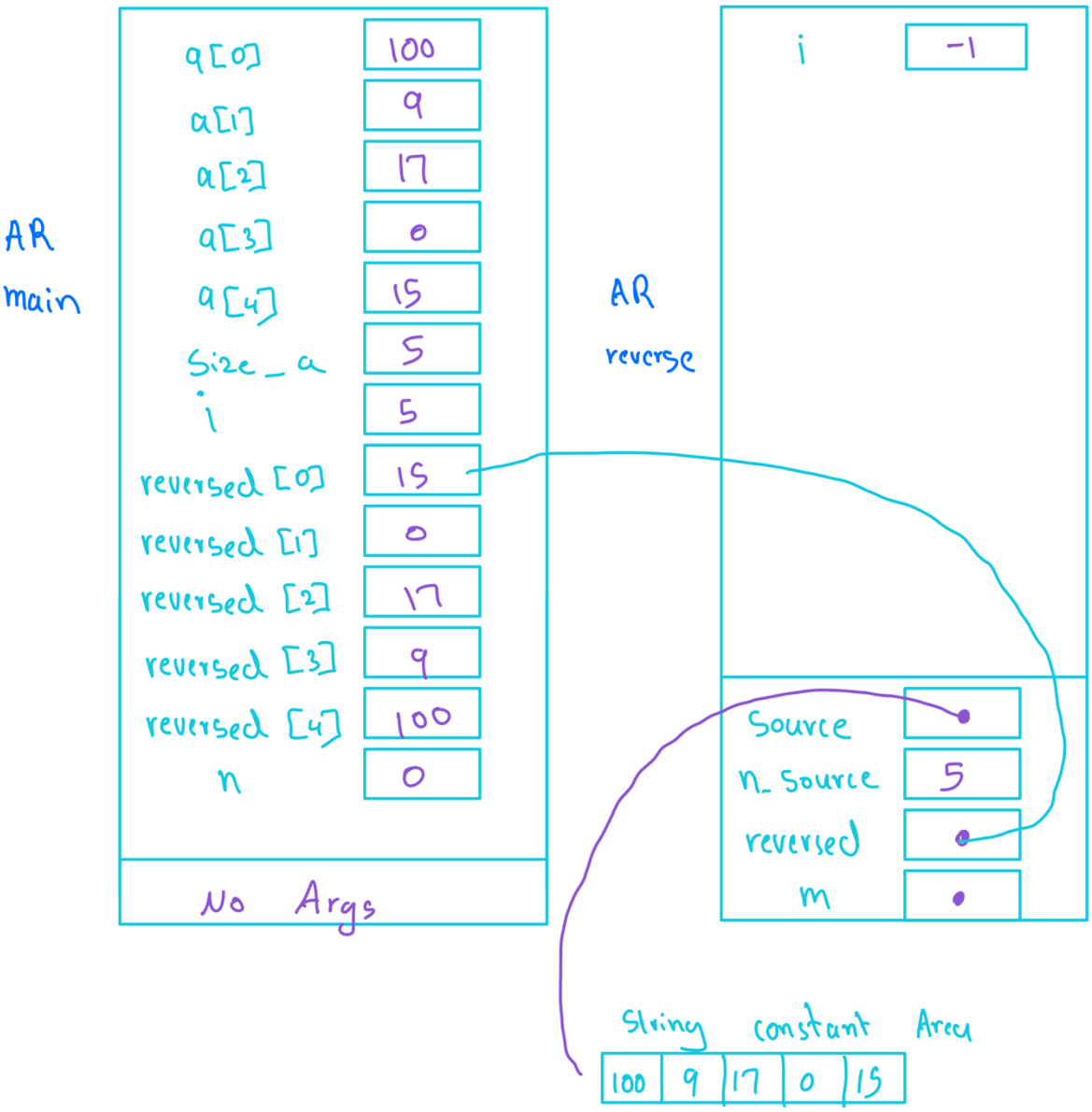


point 4

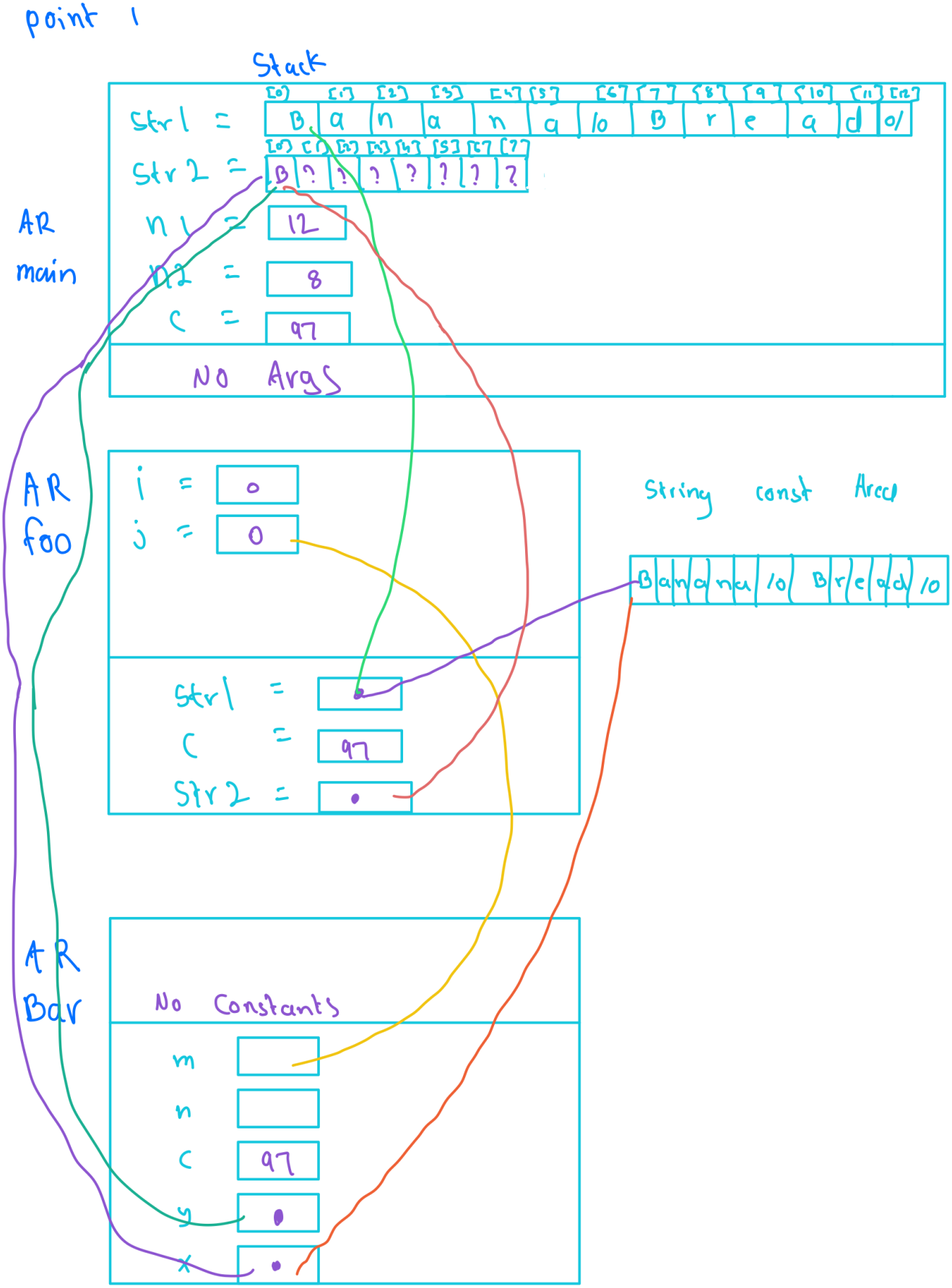


Exercise B

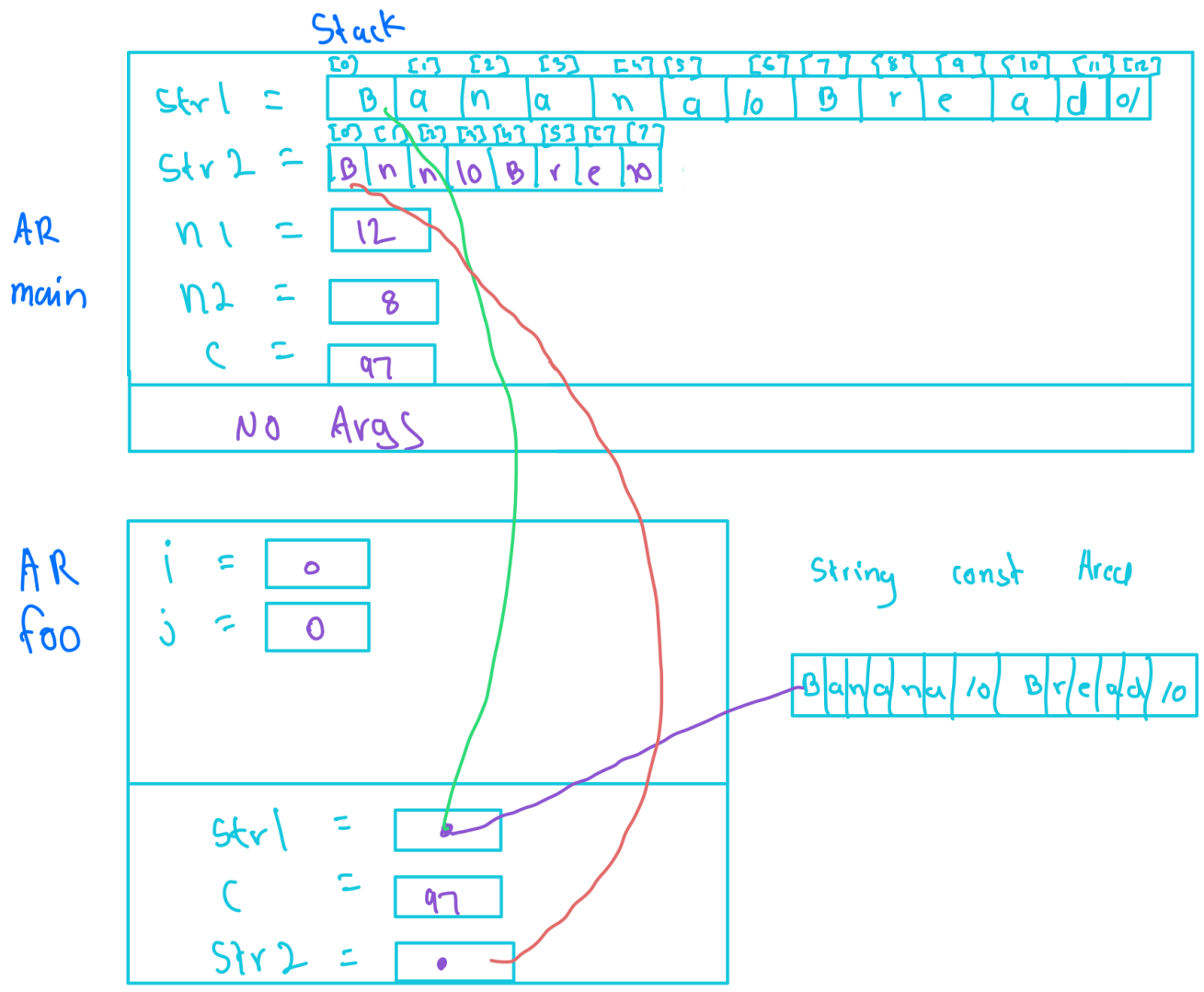
point 1



Exercise C


point 1
Stack
Str1 =
Str2 =
n1 =
n2 =
c =
AR
main
12
8
97
No Args
AR
foo
i =
j =
0
0
Str1 =
c =
Str2 =
97
String const Area
AR
Bar
No Constants
m
n
c
y
x
97

Point 2


Stack
Str1 =
Str2 =
n1 =
n2 =
c =
12
8
97
No Args
AR main
AR foo
i =
j =
0
0
Str1 =
c =
Str2 =
97
String const Area

Exercise D

main.c

test 〉 test 〉 main.c 〉 main()

```c
/*
 *  lab3exe_D.c
 *  ENSF 337, lab3 Exercise D
 *
 *  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) {
    // STUDENTS MUST COMPLETE THE REST OF IMPLEMENATION OF THIS FUNCTION
    int tri[n][n];
    for (int i = 0 ; i < n; i++){
        for (int j = 0 ; j <= i; j++){
            if (j == 0 || j == i){
                tri[i][j] = 1;
            }
            else{
                tri[i][j] = tri[i-1][j-1] + tri[i-1][j];
            }
        }
    }

    for (int i = 0 ; i < n; i++){
        printf("Row %d:\t", i);
        for (int j = 0 ; j <= i; j++){
            printf("%d\t",tri[i][j]);

        }
        printf("\n");
    }
}

```

```
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
Program ended with exit code: 0
```

Exercise E

test 〉 test 〉 C main.c 〉 No Selection

```c
/* lab3exe_E.c
 * ENSF 337, Lab 3 Exercise E
 */

#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;
}

```

```c
    return 0;
}

int substring(const char *s1, const char* s2)
{
    // This function is incomplete. Student must remove the next line and
    // complete this function...
    int i=0,j=0;
    while ( ( *(s1+i) != '\0') && ( *(s2+j) != '\0') ) {
        if(*(s1+i) == *(s2+j) ){
            j++;
            if (*(s2+j) == '\0'){
                return 1;
            }
        }
        else{
            j = 0;
        }
        i++;
    }
        return 0;
}

void select_negatives(const int *source, int n_source,
                      int* negatives_only, int* number_of_negatives)
{
    // This function is incomplete. Student must remove the next line and
    // complete this function...
    int j = 0;
    *number_of_negatives = 0;
    for (int i =0 ; i < n_source; i++){
        if ( *(source + i) < 0  ){
            *(negatives_only +j) = *(source + i);
            j++;
            *number_of_negatives = *number_of_negatives +1;

        }
    }
    return;
}
```

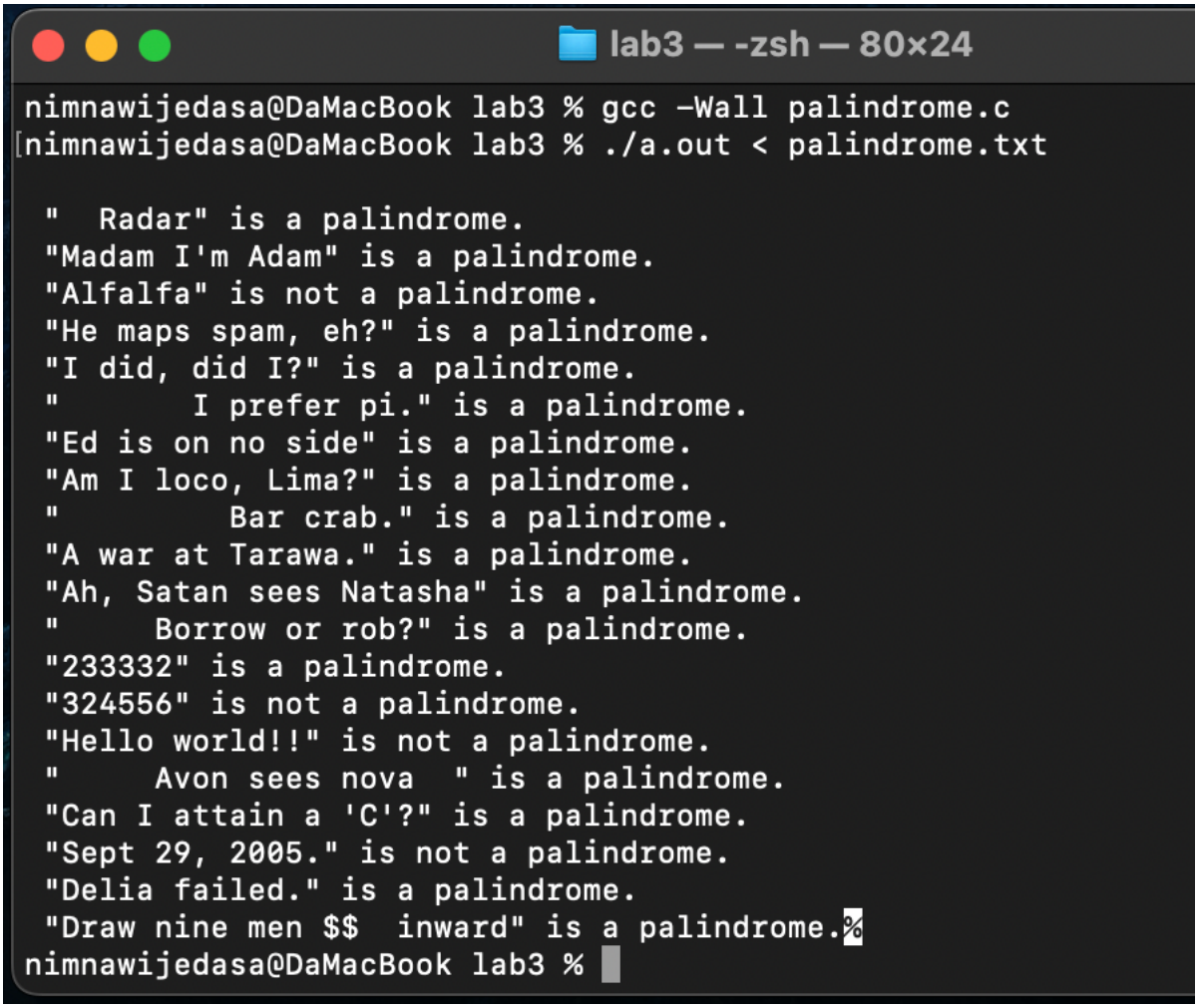
```
a has 5 elements:  -10  9  -17  0  -15

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
Program ended with exit code: 0
```

Exercise F

palindrome.c 〉 No Selection

```c
/* 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
 */

#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
```

```c

        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;
}

int is_palindrome (const char *str){
    int l = 0;
    int h = strlen(str) - 1;
    while (h > l) {
        if (str[l++] != str[h--]) {
            return 0;
        }
    }
    return 1;
}


void strip_out(char *str){
    int i=0,m=0,j=0, c=0 ;
    int len = strlen(str);
    char temp[len];

    while ( m< len){
        if ( isalnum(str[m]) ){
            temp[i] = str[m];
            c++;
            i++;
        }
        m++;
    }
    while ( j< len){
        str[j] = '\0';
        j++;
    }
    for (int k = 0; k < c ; k++){
        str[k] = tolower(temp[k])  ;
    }
}

```


lab3 — -zsh — 80×24
nimnawijedasa@DaMacBook lab3 % gcc -Wall palindrome.c
nimnawijedasa@DaMacBook lab3 % ./a.out < palindrome.txt
" 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.%
nimnawijedasa@DaMacBook lab3 %