Lab 4

Name: Pattabhiram Karthik Jonnalagadda
Lab section: B02
Course: ENCM 335
UCID: 30247339
Assignment: Lab 4

Exercise B:

code:

```c
#include <stdio.h>
#include <stdlib.h>


double get_double_or_exit(void);
// REQUIRES:
//    User has been prompted to enter a double.
// PROMISES:
//    Function tries to read a double using scanf and "%d"
//    On success, that double is echoed to the user,
//    and the double is the function return value.
//    On failure, and error message is printed and
//    exit is called with an argument of 1.
int get_int_or_exit(void);
// REQUIRES:
//    User has been prompted to enter an int.
// PROMISES:
//    Function tries to read an int using scanf and "%d"
//    On success, that int is echoed to the user,
//    and the int is the function return value.
//    On failure, and error message is printed and
//    exit is called with an argument of 1.

void get_input(double *km_in, int *minutes_in, double *seconds_in);
// PROMISES:



int main(void) {
    double km, seconds, time_s, avg_1k, avg_1m, min, sec;
    int minutes;
```

```
    get_input(&km, &minutes, &seconds);

    printf("Distance run: %f km.\n",km);

    time_s = minutes*60 +seconds;
    avg_1k = time_s / km;
    avg_1m = time_s / (km * 0.621371);

    printf("Total time of run: %d minute(s), %f second(s)\n", minutes,
seconds);
    min = (int)avg_1k / 60;
    sec = avg_1k - min*60;
    printf("Average time per 1 km: %d minute(s), %f second(s)\n", (int)min,
sec);
    min = (int)avg_1m / 60;
    sec = avg_1m - min*60;
    printf("Average time per 1 mile: %d minute(s), %f second(s)\n",
(int)min, sec);
    return 0;



};

int get_int_or_exit(void)
{
  int result;
  if (1 != scanf("%d", &result)) {
    printf("I could not read an int. I am quitting.\n");
    exit(1);
  }
  printf("I read an int value of %d.\n", result);
  return result;
}

double get_double_or_exit(void) {
  double result;
  if (1 != scanf("%lf", &result)) {
    printf("I could not read a double. I am quitting.\n");
    exit(1);
  }
  printf("I read a double value of %lf.\n", result);
  return result;
```

```
}

void get_input(double *km_in, int *minutes_in, double *seconds_in) {

    printf("Please enter a distance in km, using type double.\n");
    *km_in = get_double_or_exit();

    printf("Please enter a number of minute, using type int.\n");
    *minutes_in = get_int_or_exit();

    printf("Please enter a number of seconds, using type double.\n");
    *seconds_in = get_double_or_exit();



}
```

Output:

```
❯ ./b
Please enter a distance in km, using type double.
21.0975
I read a double value of 21.097500.
Please enter a number of minute, using type int.
89
I read an int value of 89.
Please enter a number of seconds, using type double.
45.6
I read a double value of 45.600000.
Distance run: 21.097500 km.
Total time of run: 89 minute(s), 45.600000 second(s)
Average time per 1 km: 4 minute(s), 15.271952 second(s)
Average time per 1 mile: 6 minute(s), 50.820511 second(s)
❯ ./b
Please enter a distance in km, using type double.
3.218688
I read a double value of 3.218688.
Please enter a number of minute, using type int.
7
I read an int value of 7.
Please enter a number of seconds, using type double.
54.1
I read a double value of 54.100000.
Distance run: 3.218688 km.
Total time of run: 7 minute(s), 54.100000 second(s)
Average time per 1 km: 2 minute(s), 27.296041 second(s)
Average time per 1 mile: 3 minute(s), 57.050073 second(s)
❯ ./b
Please enter a distance in km, using type double.
a
I could not read a double. I am quitting.
❯ ./b
Please enter a distance in km, using type double.
12
I read a double value of 12.000000.
Please enter a number of minute, using type int.
a
I could not read an int. I am quitting.
❯ ./b
Please enter a distance in km, using type double.
12
I read a double value of 12.000000.
Please enter a number of minute, using type int.
12
I read an int value of 12.
Please enter a number of seconds, using type double.
aa
I could not read a double. I am quitting.

 ~/desktop/ENCM 335/labs/lab4 on  main ?1 ····· took  5s at  06:04:04 PM
❯
```

Exercise D:

Code:

```
// ENCM 335 Fall 2025 Lab 4 Exercise D

#include <stdio.h>

int main(void)
{
  char buffer[80];   // enough space for a string of length <= 79

  // THIS IS A GOOD WAY TO LEARN SOMETHING ABOUT C STRINGS, BUT IT'S
  // NOT A GOOD EXAMPLE OF READABLE OR PRACTICAL CODE!

  // Put characters into the string using ASCII codes.
  // make it so that buffer contains the string "In C the value of 200 % (6
+ 3) == 180 is 1."
  buffer[0] = 73;   // 'I'
  buffer[1] = 110;  // 'n'
  buffer[2] = 32;   // ' '
  buffer[3] = 67;   // 'C'
  buffer[4] = 32;   // ' '
  buffer[5] = 116;  // 't'
  buffer[6] = 104;  // 'h'
  buffer[7] = 101;  // 'e'
  buffer[8] = 32;   // ' '
  buffer[9] = 118;  // 'v'
  buffer[10] = 97;  // 'a'
  buffer[11] = 108; // 'l'
  buffer[12] = 117; // 'u'
  buffer[13] = 101; // 'e'
  buffer[14] = 32;  // ' '
  buffer[15] = 111; // 'o'
  buffer[16] = 102; // 'f'
  buffer[17] = 32;  // ' '
  buffer[18] = 50;  // '2'
  buffer[19] = 48;  // '0'
  buffer[20] = 48;  // '0'
  buffer[21] = 32;  // ' '
  buffer[22] = 37;  // '%'
  buffer[23] = 32;  // ' '
  buffer[24] = 40;  // '('
  buffer[25] = 54;  // '6'
  buffer[26] = 32;  // ' '
  buffer[27] = 43;  // '+'
```

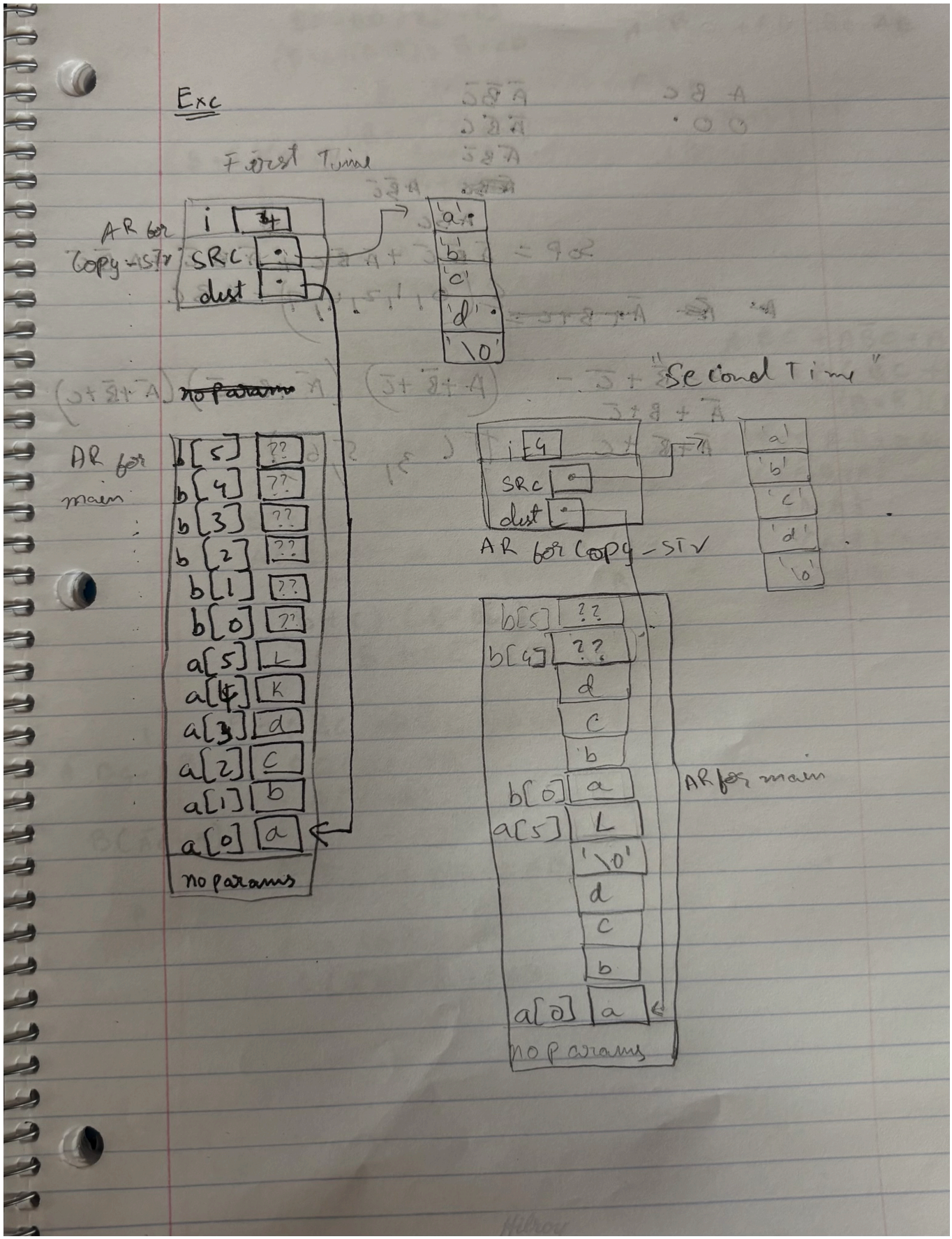
```
    buffer[28] = 32;  // ' '
    buffer[29] = 51;  // '3'
    buffer[30] = 41;  // ')'
    buffer[31] = 32;  // ' '
    buffer[32] = 61;  // '='
    buffer[33] = 61;  // '='
    buffer[34] = 32;  // ' '
    buffer[35] = 49;  // '1'
    buffer[36] = 56;  // '8'
    buffer[37] = 48;  // '0'
    buffer[38] = 32;  // ' '
    buffer[39] = 105; // 'i'
    buffer[40] = 115; // 's'
    buffer[41] = 32;  // ' '
    buffer[42] = 49;  // '1'
    buffer[43] = 46;  // '.'

    // Put the end-of-string character at the end of the string.
    buffer[44] = 0;

    printf("The string in buffer is \"%s\"\n", buffer);
    return 0;
}
```

Exercise C:

AR diagram:



Exercise E:

Code:

```c
// ENCM 335 Fall 2025 Lab 4 Exercise E

#include <stdio.h>
#include <string.h>

int safecat(char *dest, const char* src, int dest_size);
// REQUIRES
//    dest_size > 0.
//    Array elements dest[0] ... dest[dest_size-1] exist.
//    dest points to the beginning of a C string with
//    length < dest_size.
//    src points to the beginning of a C string.
// PROMISES
//    If the sum of the lengths of the two strings is less than
//    dest_size, the string in the dest array is the concatenation
//    of the original string in dest and the string from src, and
//    the return value is the length of the new string.
//
//    If not, the string in dest is the concatenation of the original
//    string in dest and as many characters from src as possible,
//    while leaving room for '\0' in dest[dest_size-1], and the return
//    value is -1.

int main(void)
{
  int rv;
  char buf[10];

  buf[0] = '\0';
  rv = safecat(buf, "", 10);
  printf("buf contains \"%s\" and rv is %d (expect \"\" and 0).\n",
         buf, rv);
  rv = safecat(buf, "0123", 10);
  printf("buf contains \"%s\" and rv is %d (expect \"0123\" and 4).\n",
         buf, rv);
  rv = safecat(buf, "45678", 10);
  printf("buf contains \"%s\" and rv is %d (expect \"012345678\" and 9).\n",
         buf, rv);

  buf[0] = '\0';
  rv = safecat(buf, "abcde", 10);
  printf("buf contains \"%s\" and rv is %d (expect \"abcde\" and 5).\n",
         buf, rv);
  rv = safecat(buf, "fghij", 10);
  printf("buf contains \"%s\" and rv is %d (expect \"abcdefghi\" and
-1).\n",
```

```
          buf, rv);

  buf[0] = '\0';
  rv = safecat(buf, "01", 10);
  printf("buf contains \"%s\" and rv is %d (expect \"01\" and 2).\n",
          buf, rv);
  rv = safecat(buf, "2345", 10);
  printf("buf contains \"%s\" and rv is %d (expect \"012345\" and 6).\n",
          buf, rv);
  rv = safecat(buf, "6789ABCDEF", 10);
  printf("buf contains \"%s\" and rv is %d (expect \"012345678\" and
-1).\n",
          buf, rv);

  return 0;
}

int safecat(char *dest, const char* src, int dest_size)
{
       int dest_len =0 , src_size =0;

       for (int i = 0; src[i] !='\0'; i++, src_size++);
       for (int i = 0; dest[i] !='\0'; i++, dest_len++);


       if ((dest_len + src_size) < dest_size) {
              for (int i=0; i< src_size; i++) {
                     dest[i+dest_len] = src[i];
              }
              dest[src_size+dest_len] = '\0';

       return dest_len+src_size;
       } else {
              for (int i=0; i< (dest_size - dest_len - 1); i++) {
                     dest[i+dest_len] = src[i];
              }
              dest[dest_len + (dest_size - dest_len - 1)] = '\0';
              return -1;
       }
}
```

Output:

```
❯ gcc lab4exe.c -o e
❯ ./e
buf contains "" and rv is 0 (expect "" and 0).
buf contains "0123" and rv is 4 (expect "0123" and 4).
buf contains "012345678" and rv is 9 (expect "012345678" and 9).
buf contains "abcde" and rv is 5 (expect "abcde" and 5).
buf contains "abcdefghi" and rv is -1 (expect "abcdefghi" and -1).
buf contains "01" and rv is 2 (expect "01" and 2).
buf contains "012345" and rv is 6 (expect "012345" and 6).
buf contains "012345678" and rv is -1 (expect "012345678" and -1).
```

#encm335