Exercise C

Code:

```
/*
 *  lab3exe_C.c
 *  ENSF 337, lab3 Exercise C
 *  Jiho Kim
 *  In this program the implementation of function pascal triangle is missing.
 *  Student 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 i = 0, j=0;
            int p[n][n];

            while(i<n)
            {
                    while(j<=i){
                            if(j == 0 || j == i){
                                    p[i][j++] = 1;
                            } else{
                                    p[i][j++] = p[i-1][j-1] + p[i-1][j];
                            }
                    }

                    i++;
                    j = 0;
```

```
        }


        i=0;
        while(i<n){
                while(j<=i)
                {
                        printf("%d   ", p[i][j++]);
                }
                j=0;
                i++;
                printf("\n");
        }

}
```

OUTPUT:

```
Enter the number of rows (Max 20): 9
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10   10   5   1
1   6   15   20   15   6   1
1   7   21   35   35   21   7   1
1   8   28   56   70   56   28   8   1
```

Exercise D

Code:

```
/* lab3exe_D.c
 * ENSF 337, Lab 3 Exercise D
 * Jiho Kim
 */

#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)
{
    // This function is incomplete. Student must remove the next line and
    // complete this function...
    //printf ("\nFunction substring is incmplete and doesn't work.\n");
    int i = 0, j=0, z=0;
    //note i will be used as a counter for s1, j will be used as counter for s2, z will be used as a
placeholder.
    int x = strlen(s1), y = strlen(s2);
    while(i < x){
        if(s1[i] == s2[j]){
            z = i;
            while(s1[i] == s2[j]){
                //printf("%c", s1[i]);
                if(j == (y-1)){
                    return 1;
                }
                i++;
                j++;

            }
            i = z;
            j = 0;
        }
        i++;
    }
    if(i == x)
        return 0;


    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...
    //printf ("\nFunction select_negatives is incmplete and doesn't work.\n");

    int i = 0, j = 0;
    *number_of_negatives = 0;

    while(i < n_source){
        if(source[i] < 0)
        {
            negatives_only[j] = source[i];
            j++;
        }


        i++;
    }
    *number_of_negatives = j;
    return;
}
```

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 E

Code:

```
/* File: palindrome.c
 *  ENSF 337
 *  Exercise E - 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);
        //printf("%s", 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)
{
    int i = 0, j = 0;
    //i will be counter of original str and j will be counter of counter to keep track of changed str
    while(i<strlen(str))
    {
        if( (str[i] >= 'a' && str[i] <= 'z') || (str[i] >= '0' && str[i] <= '9') || (str[i] >= 'A' && str[i] <= 'Z'))
        {
            str[j] = str[i];
            i++;
            j++;
        } else{
            i++;
        }
    }
    str[j] = '\0';


    return;
}


int is_palindrome (const char *str)
{
    strlwr(str);
//note that this line makes the entire string to lower case.
```

```
//if desired, you can use a loop to go through each individual characters and change if it is an uppercase.
    int i = 0, j = strlen(str)-1;

    while(i <= j){
        if(str[i] == str[j]){
            i++;
            j--;
        } else {
            break;
        }
    }

    if(i >= j){
        return 1;
    }
    return 0;

}
```

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.
```