|  |
| --- |
| **Course**: Programming Fundamental – ENSF 337  **Lab #**: Lab 3  **Instructor**: M. Moussavi  **Student Name**: Aarushi Roy Choudhury  **Lab Section**: B01  **Date submitted**: Oct, 14 2021 |

**Exercise A**

Text

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidenceA picture containing table

Description automatically generated

Diagram

Description automatically generatedText

Description automatically generated with low confidence

**Exercise B**

Diagram

Description automatically generated

**Exercise C**

**A picture containing icon

Description automatically generated**

Diagram

Description automatically generated

**Text, whiteboard

Description automatically generated**Diagram

Description automatically generated

**Exercise D**

#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 arr[n][n];

   int i=0,j=0;

   for(i=0;i<n;i++){

       for(j=0;j<=i;j++){

            if(j==0 || j==i)

                arr[i][j]=1;

           else

               arr[i][j]=arr[i-1][j-1]+arr[i-1][j];

       }

   }

   for(i=0;i<n;i++){

       printf("row %d:",i );

       for(j=0;j<=i;j++){

           printf("   %d",arr[i][j] );

       }

       printf("\n");

   }

}

Text

Description automatically generated

**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;

}

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

{

    int i,j,k;

    for(i=0;s1[i] !='\0';i++){

        for(j=i,k=0;s2[k]!='\0'&& s1[j]==s2[k];j++,k++){

            ;

        }

        if(k >0 && s2[k]=='\0')

          return 1;

    }

     return 0;

}

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

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

{

    int i;

    \*number\_of\_negatives = 0;

     int j=0;

    for(i=0;i<n\_source;i++){

        if(source[i]<0){

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

            j++;

        }

    }

    \*number\_of\_negatives = j;

    return ;

}

Text

Description automatically generated

**Exercise F**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define SIZE 100

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);

    if (str[strlen(str) - 1] == '\n')

        str[strlen(str) - 1] = '\0';

    strcpy(temp,str);

    while(strcmp(str, "done") !=0)

    {

#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;

}

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

    int i = 0;

    int j = strlen(str) - 1;

    char x;

    char y;

    while(j > i){

        x = str[i];

        y = str[j];

        if(isupper(x)){

            x = tolower(x);

        }

        if(isupper(y)){

            y = tolower(y);

        }

        if(x != y){

            return 0;

        }

        ++i;

        --j;

    }

    return 1;

}

void strip\_out(char \*str){

    char \*p;

    char copy[100];

    int len = 0;

    for (p = str; \*p != '\0'; p++) {

        if(isalnum(\*p)){

           copy[len] = \*p;

           ++len;

        }

    }

    copy[len] = '\0';

    strcpy(str, copy);

}

Text

Description automatically generated