**Student Name: Codrin Oneci**

**16.35 Pset 1 Written Solution**

**Question 1**

1. **Introduction**

This document is a SRS for the 16.35 2D flight simulator.

The 2D flight simulator is a software product used to simulate multiple point-like aircraft using various control policies.

1. **Overall description**
   1. The 2D flight simulator will be self contained and independent of all other software.

1. **Specific requirements**
   1. **Clock**
      1. The 2D flight simulator shall have a simulator clock;
   2. **Aircraft** 
      1. The aircraft shall be described as an Abstract Data Type (ADT) by the following parameters: heading direction, x coordinate on the map, y coordinate on the map, control policy;
      2. The aircraft shall be treated as a point object with an orientation at any clock time;
      3. The aircraft parameters shall be settable by external parameters;
      4. The aircraft dynamics shall be described by linear and angular velocities;
      5. The aircraft’s linear and angular velocities shall be limited to predefined domains;
      6. An aircraft object shall not collide with any other aircraft in the same simulation (collisions shall be neglected);
      7. The aircraft shall not have x and y coordinates outside a predefined map domain corresponding to each simulation. The map domain shall have no obstacles and shall be represented solely by its borders.
   3. **Control Policy**
      1. The 2D flight simulator shall have multiple control policies
      2. Polygon Control Policy
         1. The 2D flight simulator shall have a control law which guides the aircraft to follow a regular polygon;
         2. The Polygon Control Policy shall be described by a number of sides of the polygon and the diameter of the circle circumscribing the polygon;
         3. The polygon number of sides and the circle diameter shall be set by external parameters;

**Question 2**

//Author: Codrin Oneci

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(int argc, char\*\* argv) {

double x,y, out\_value;

char oper;

if(argc!=4) //I expect only 3 arguments (3 strings) but I have to count the program name too so 3+1=4

{

printf("Invalid Input\n");

return -1; //this is optional C convention in which return 0 means ok while return -1 will suggest error

}

//obtain the values for the numbers; argv[i] gives string array at location number i using the pointer

x = atof(argv[1]); //convert string to double as sugested in the pset text

y = atof(argv[3]); //convert string to double

//Important note: if the command line inputs for x or y are actually not

//numeric, the atof function outputs 0.0 so check this possibility

int j=0,i=0,dot\_count=0;

j=strlen(argv[1]);

if (x==0)

{

for (i=0;i<j;i++){

if (argv[1][i]!=48&&argv[1][i]!="."&&argv[1][i]!='\0')

{

printf("Invalid Input\n");

return -1;

}

if (argv[1][i]==".")

{

dot\_count++;

}

}//48 is the ASCII value of 0; and '\0' is the termination character

}

if (dot\_count>1)

{

printf("Invalid Input\n");

return -1;

}

dot\_count=0;//put the counter to zero

j=strlen(argv[3]);

if (y==0)

{

for (i=0;i<j;i++){

if (argv[3][i]!=48&&argv[3][i]!="."&&argv[1][i]!='\0')

{

printf("Invalid Input\n");

return -1;

}

if (argv[3][i]==".")

{

dot\_count++;

}

}

}

if (dot\_count>1){printf("Invalid Input\n");return -1; }

//extract the operation value

oper=argv[2][0];

switch(oper)

{

case '\*':

out\_value=x\*y;

break;

case '/':

out\_value=x/y;

break;

case '-':

out\_value=x-y;

break;

case '+':

out\_value=x+y;

break;

case 'x':

out\_value=x\*y;

break;

case 'X':

out\_value=x\*y;

break;

default:

printf("The operation type input is not recognized. Choose from [\*,/,-,+,x,X].");

break;

}

if(oper=='\*' || oper=='/' || oper=='-' || oper=='+' || oper=='x' || oper=='X' )

{

printf("%f",out\_value);

}

else

{

return -1 ;

}

return 0;

}

**Question 3**

//Author: Codrin Oneci

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

int main(int argc, char\*\* argv)

{ int p=1;//assume symmetry

int count\_characters=0; //this variable will store a count of the total number of characters

if (argc<=1)

{ printf(" is NOT a palindrome!\n");

return 0;}//In the case that the program does not receive at least an input string (word)

for (int i=1; i<argc; i++)

{

count\_characters+=strlen(argv[i]);

}

char \*concat\_str=malloc(count\_characters);//in this array I concatenate all the words

for (int i=1; i<argc; i++)

{

strcat(concat\_str,argv[i]);

}

//at this point, I have stored all the relavant data in a single char array called concat\_str

int l=strlen(concat\_str);//length of the string with concatenated words

int i=0;

//create a lowercase string now from the initial string of concatenated words

//observe that in the example that you gave us the program is case insensitive

char \*concat\_str\_l=malloc(count\_characters);

while(concat\_str[i])

{

concat\_str\_l[i]=tolower(concat\_str[i]);

i++;

}

for(i = 0; i < l; i++)

{if(concat\_str\_l[i] != concat\_str\_l[l - i - 1]) //check palindrome symmetry

{

p = 0;

break;

} //if not symmetric, it is not a palindrome so p=0

}

char \*input\_str=malloc(count\_characters+argc);//in this array I concatenate all the words with spaces too for the printing

for (int i=1; i<argc; i++)

{ strcat(input\_str,argv[i]);

if (i<argc-1)

{

strcat(input\_str," ");

}

}

if (p==1)

printf("\"%s\" is a palindrome!\n",input\_str);

else

printf("\"%s\" is NOT a palindrome!\n",input\_str);

return 0;

}