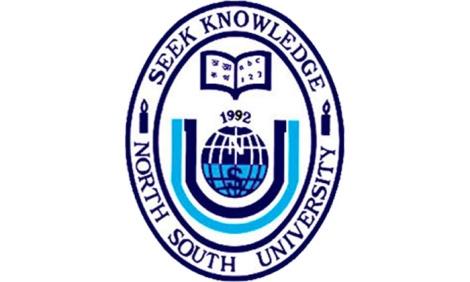
**CSE115L Sec-25: Programming Language I Lab  
 Final Examination Assignment, Spring 2020  
   
 Date Posted: Monday, May 18, 2020  
 Date Due: Thursday, June 04, 20  
 Total Marks Allotted: 70 Total Questions: 05**

**Q1**. **Password Checker**

**Write** a C program that will take a string called password as input and will print a message indicating whether it is a Strong Password or a Weak Password.

**N.B:** The program will keep asking the user for a password until the user enters a Strong Password.

A password is Strong if it fulfills the following properties:

1. It should have at least 8 characters.
2. It should have at least one lower case letter and one upper case letter.
3. It should not contain any blank space.
4. It must contain a number.

[**Marks:]**

**Sample Input 1:**

Enter the password: kobe8

**Sample Output 1:**

Weak Password! Please enter again: Kobe808

Weak Password! Please enter again: Kobe8080

Strong Password!

**Sample Input 2:**

Enter the password: KobeBryant24

**Sample Output 2:**

Strong Password!

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <stdbool.h>  void main()  {  bool has\_lower = false;  bool has\_upper = false;  bool has\_space = false;  bool has\_number = false;  bool strong\_pass = false;  char password[25];  printf("Enter the password: ");  while(strong\_pass == false){  gets(password);  if(strlen(password) >= 8){  for(int i=0; i<strlen(password); i++){  if(password[i] >= 'a' && password[i] <= 'z'){  has\_lower = true;  }  if(password[i] >= 'A' && password[i] <= 'Z'){  has\_upper = true;  }  if(password[i] == ' '){  has\_space = true;  }  if(password[i] >= '0' && password[i] <= '9'){  has\_number = true;  }  }  if(has\_lower && has\_upper && !has\_space && has\_number){  strong\_pass = true;  }  }  if(strong\_pass == true){  printf("\nStrong Password! ");  }  else{  printf("\nWeak Password! Please enter again: ");  }  }  } |

**Q2. Polygon Area Calculator**

**Write** a C program along with a function called double area (int n, double side**)** that returns the area of a regular polygon.The function will take two parameters: the number of sides, n and the length of each side, side. Write a main method that prompts the user to enter the number of sides and the side of a regular polygon and displays its area.

N.B: You do not need to write the whole program, just write the function.

[**Marks:]**

**Definition:**

A regular polygon is an n-sided polygon in which all sides are of the same length and all angles have the same degree (i.e., the polygon is both equilateral and equiangular). The formula for computing the area of a regular polygon is

Here, is the number of sides in a polygon, and is the length of each of the sides.

**Input**

The first input will be an integer preceded by the following texts: “Enter the number of sides”. Then the user will proceed to give input of number of sides in the polygon. The second input will be a double preceded by the following texts: “Enter the side”. Then the user will proceed to give input for the length of each side.

**Output**

The first output will be a double preceded by the following texts: “The area of the polygon is”. Then the program will calculate the area of the polygon. The output will be printed in the main function, your area function will just return the value.

**Sample Input 1**

Enter the number of sides: 5

Enter the side: 6.5

**Sample Output 1**

The area of the polygon is 72.69017017488385 [**This line will be printed inside the main function**]

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <math.h>  double area(int n, double side){  return (n \* side \* side) / (4 \* tan(M\_PI/n));  }  void main(){  int n;  double side;  double result;  printf("Enter the number of sides: ");  scanf("%d",&n);  printf("Enter the side: ");  scanf("%lf",&side);  result = area(n,side);  printf("The area of the polygon is %lf", result);  } |

**Q3. Sum of Array**

**Write a C program** which will create an integer array of size given by the user and ask the user to enter the values in it. Then write a function named int sum (int \*ptr, int size)that computes and returns the sum of the elements of the array. But you cannot pass the whole array as parameter, rather you can just pass the pointer ptr and the size of the array called size, which contains the base address of the array. Also, if any number in the array is divisible by 5 then the function should skip that number while calculating the sum. Memory must be allocated using malloc() and must be released using free().

[**Marks:]**

**Sample Input 1**

Enter the size of the array: 6

Enter the elements of the array:

1  
5  
-3  
2  
4  
15

**Sample Output 1**

The sum of the array (excluding the numbers that are divisible by 5) is: 4

**Sample Input 2**

Enter the size of the array: 4

Enter the elements of the array:

19  
5  
-10  
4

**Sample Output 2**

The sum of the array (excluding the numbers that are divisible by 5) is: 23

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  int sum(int \*ptr, int size){  int sum = 0;  for(int i=0; i<size; i++){  if(ptr[i] % 5 != 0){  sum += ptr[i];  }  }  return sum;  }  void main(){  int size;  int result;  printf("Enter the size of the array: ");  scanf("%d",&size);  int \*arr = malloc (sizeof (int) \* size);  printf("\nEnter the elements of the array: \n");  for(int i=0; i<size; i++){  scanf("\n%d",&arr[i]);  }  result = sum(arr,size);  printf("The sum of the array (excluding the numbers that are divisible by 5) is: %d", result);  } |

**Q4. Shortest Line Calculator**

**Write** a C program which will calculate the length of lines using end-points of each line, and determine the lines with highest length and lowest length. The program should contain following structures:

Create a structure named Point:

1. int
2. int

Create a structure named Line:

1. Point firstEndPoint
2. Point secondEndPoint

Also, create an array of Line with size , which will be taken from user and find out the lines with highest and lowest length. .

[**Marks:]**

**Description**

Distance of a Line, , where d equals the distance of the line, equal the coordinates of the first endpoint of the line segment, and equal the coordinates of the second endpoint of the line segment.

**Input**

Input is an integer preceded by the following texts: “Enter number of lines:”. Then the user will proceed to give input of integer of the number of lines.

The next input is two integers preceded by the following texts: “Enter first end point of line 1:”. Then the user will proceed to give input of integers of the first endpoint of a line.

The next input is two integers preceded by the following texts: “Enter second end point of line 1:”. Then the user will proceed to give input of integers of the second endpoint of a line.

**Output**

The first output is a double preceded by the following texts: “Highest length of a line is:” following by the output of highest length among the given lines.

The second output is a double preceded by the following texts: “Lowest length of a line is:” following by the output of lowest length among the given lines.

**Sample Input 1**

Enter number of lines: 2  
Enter first end point of line 1: 2 5  
Enter second end point of line 1: 3 4  
Enter first end point of line 2: 2 6  
Enter second end point of line 2: 3 8

**Sample Output 1**

Highest length of a line is: 4.1231  
Lowest length of a line is: 1.4142

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  struct Point  {  int x1, y1;  };  struct Line  {  struct Point firstEndPoint;  struct Point secondEndPoint;  };  void mainShortestLineCalculator(){  int n = 0;  int x\_square = 0;  int y\_square = 0;  double distance = 0;  double highest\_length = 0;  double lowest\_length = 0;  int lowest\_init\_flag = 0;  printf("Enter number of lines: ");  scanf("%d",&n);  struct Line LineArr[n];  for(int i=0; i<n; i++){  printf("Enter first end point of line %d: ", i+1);  scanf("%d%d", &LineArr[i].firstEndPoint.x1, &LineArr[i].firstEndPoint.y1);  printf("Enter second end point of line %d: ", i+1);  scanf("%d%d", &LineArr[i].secondEndPoint.x1, &LineArr[i].secondEndPoint.y1);  //Distance  x\_square = (LineArr[i].secondEndPoint.x1 - LineArr[i].firstEndPoint.x1) \* (LineArr[i].secondEndPoint.x1 - LineArr[i].firstEndPoint.x1);  y\_square = (LineArr[i].secondEndPoint.y1 - LineArr[i].firstEndPoint.y1) \* (LineArr[i].secondEndPoint.y1 - LineArr[i].firstEndPoint.y1);  distance = sqrt(x\_square + y\_square);  //highest  if(distance > highest\_length){  highest\_length = distance;  }  //lowest  if(lowest\_init\_flag == 0){  lowest\_length = distance;  lowest\_init\_flag = 1;  }  else{  if(distance < lowest\_length){  lowest\_length = distance;  }  }  }  printf("Highest length of a line is: %lf\n", highest\_length);  printf("Lowest length of a line is: %lf\n", lowest\_length);  } |

**Q5. Course Project**

Your course project will carry 25 marks. The group project is already selected, and I hope you have started working on it. The submission date of the project will be announced later in Google Classroom, but it won’t be before June 04. The marks distribution of the project is the following:

|  |  |  |
| --- | --- | --- |
| **Part** | **Marks** | **Group/Individual Marking?** |
| Project | 15 | Group |
| Project Report | 5 | Group |
| Project Presentation | 10 | Individual |

[**Marks:]**

*The end*