غلام اسحٰق خان انسٹیٹیوٹ آف انجینیرنگ سائنس وٹیکنالوجی

Ghulam ishaq khan institute of engineering sciences and technology

Title: Computing and AI　　**Code:** CS101　　**Total Marks:** 25

Instructor: Dr. Khurram Khan　　**Project**　　**Deadline:** 27/12/2023

Q No 01

This project simulates a robot wandering around and trying to avoid bumping into anything. This is a typical robot behavior called *obstacle avoidance*. The robot has 4 *distance* sensors that point in 4 directions:

Each sensor returns a value between 0 and 100. The smaller the value, the closer the robot is to an object. The bigger the value, the further the robot is from an object. In order to avoid obstacles, the robot should always move in the direction of the sensor with the largest value.

```
          north
            ↑
west  ←     ○     →  east
            ↓
          south
```

Below is part of an obstacle avoiding robot simulation program. Your job is to fill in the missing pieces.

```
#include <stdlib.h>
#include <sys/time.h>
#include <iostream>
#include <string>
using namespace std;

// function prototypes
int senseDistance();
void getSensors( int sensors[] );
void printSensors( int sensors[], string labels[] );
int furthest( int sensors[] );

// main function
int main() {
  int sensors[4];           // array of four distance sensors
  string labels[4];         // array of labels for each sensor
  int dir;                  // indicates which direction to go next
  srand( time( NULL ));     // initialize the random number generator
  labels[0] = "north";      // initialize the sensor labels
  labels[1] = "west";
  labels[2] = "south";
  labels[3] = "east";
  for ( int i=0; i<5; i++ ) { // simulate 5 moves by the robot
    getSensors( sensors );    // get values for all the sensors
    printSensors( sensors, labels ); // print out the sensor values
    dir = furthest( sensors ); // find the direction that is furthest from an obstacle
    cout << "moving " << labels[dir] << endl; // ‘‘go’’
  }
} // end of main()
```

1. Create a file called **avoid.cpp** and type in the code above. You need to type in all the comments :-)

2. Complete the function senseDistance(), which should return a random integer between 0 and 100.
3. Complete the function getSensors(), which should assign a distance value to each element in the sensors[] array, by calling senseDistance() to get the distance value for each sensor.
4. Complete the function printSensors(), which should print out the value of each sensor, preceded by its label. For example, if sensors[0] = 93, sensors[1] = 80 sensors[2] = 73 and sensors[3] = 28, then the output might look like this:

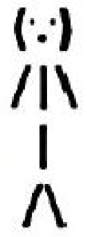
   ```
   sensors = [north=93][west=80][south=73][east=28]
   ```

   Make sure that you use the labels array and don't hardcode any labels when printing out the sensor values (i.e., output "north" by using labels[0]).

5. Complete the function furthest(), which should return the *index* of the entry in the sensors array that has the largest value. For example, given the value of sensors[] shown above, the output from furthest() should be 0 because sensors[0] is 93, which is larger than the other elements in the sensors[] array.

Q No 02

In this task you have to develop the game "Hang the Man". The concept of the game is, computer will think of any country/city, it will display underscores which will be equivalent to the length ofthe country/city name the computer has thought of. For example in case computer thinks China then it will display----- The user may enter any character as his option. If the chosen character is not in the name of city/country user looses 1 chance out of 8 possible chances of mistake and one component of human body is displayed per mistake (see the snap below). Otherwise, the chosen character is replaced in the underscore.



Note: You must NOT use any built in function for string processing.

Constraints:
1. Once a character is chosen, it cannot be chosen again.
2. Maximum number of mistakes is 11 {i.e., when the body is completed).
3. Display the number of chances left for the user
4. Clear the screen and display latest data on each iteration of user input.
5. At the end, show the result, i.e., win/loose with the correct answer that will be the name of the city/country.

Submission instructions

- You will be upload zip folder with regnumber of group memebers e.g 2020345-2020456-project.