|  |
| --- |
| **Course: Programming Fundamental – ENSF 337**  Lab #: 2  Instructor: Khedr  Student Name: Aleksander Berezowski and Kartik Sharma  Lab Section: B04  Date submitted: Sept 27th, 2021 |

**Exercise A**

Source Code:

/\*

\* File Name: lab2exe\_A.c

\* Assignment: Lab 2 Exercise A

\* Lab section: B04

\* Completed by: Kartik Sharma and Aleksander Berezowski

\* Submission Date: On or before Sept 28, 2021

\*/

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

//Global Constant Definitions

const double G = 9.8; /\* gravitation acceleration 9.8 m/s^2 \*/

const double PI = 3.141592654;

//Global Function Definitions

double degree\_to\_radian(double degrees);

double Projectile\_travel\_distance(double gravity, double v, double angle);

double Projectile\_travel\_time(double gravity, double v, double angle);

void create\_table(double velocity);

//Main Function

int main(void)

{

int n;

double velocity;

printf ("Please enter the velocity at which the projectile is launched (m/sec): ");

n = scanf("%lf" ,&velocity);

if(n != 1)

{

printf("Invlid input. Bye...");

exit(1);

}

while (velocity < 0 )

{

printf ("please enter a positive number for velocity: ");

n = scanf("%lf", &velocity);

if(n != 1)

{

printf("Invlid input. Bye...");

exit(1);

}

}

create\_table(velocity);

return 0;

}

double degree\_to\_radian(double degrees)

{

//DESCRIPTION: Converts unit "Degrees" to SI unit "Radian"

//PROMISES: Return value is a unit in radian

//REQUIRES: degrees must belong to all real numbers, fit within double type, and be in units of degrees

return (degrees/180)\*PI;

}

double Projectile\_travel\_distance(double gravity, double velocity, double angle)

{

//DESCRIPTION: Calculates the approximate distance traveled by a projectile

//PROMISES: Returns value is a distance in meters

//REQUIRES: gravity must belong to all real numbers, fit within double type, be in units of m^2/s, and not be 0

//REQUIRES: velocity must belong to all real numbers, fit within double type, and be in units of m/2

//REQUIRES: angle must belong to all real numbers, fit within double type, and be in radians

double distance = ((velocity\*velocity)/gravity)\*(sin(2\*degree\_to\_radian(angle)));

return distance;

}

double Projectile\_travel\_time(double gravity, double velocity, double angle)

{

//DESCRIPTION: Calculates the approximate travel time to maximum horizontal distance by a projectile

//PROMISES: Returns value is a time in seconds

//REQUIRES: gravity must belong to all real numbers, fit within double type, be in units of m^2/s, and not be 0

//REQUIRES: velocity must belong to all real numbers, fit within double type, and be in units of m/2

//REQUIRES: angle must belong to all real numbers, fit within double type, and be in radians

double time = (2 \* velocity \* sin(degree\_to\_radian(angle)))/gravity;

return time;

}

void create\_table(double velocity)

{

//DESCRIPTION: Creates a table of values with columns of angle, time, and distance

//PROMISES: Returns nothing, prints table of values

//REQUIRES: velocity must belong to all real numbers, fit within double type, and be in units of m/2

printf("Angle\t\tt\t\td\n(deg)\t\t(sec)\t\t(m)\n");

for (double i = 0; i<=90; i+=5)

{

double t = Projectile\_travel\_time(G,velocity,i);

double d = Projectile\_travel\_distance(G, velocity, i);

printf("%f\t%f\t%f\n",i, t, d);

}

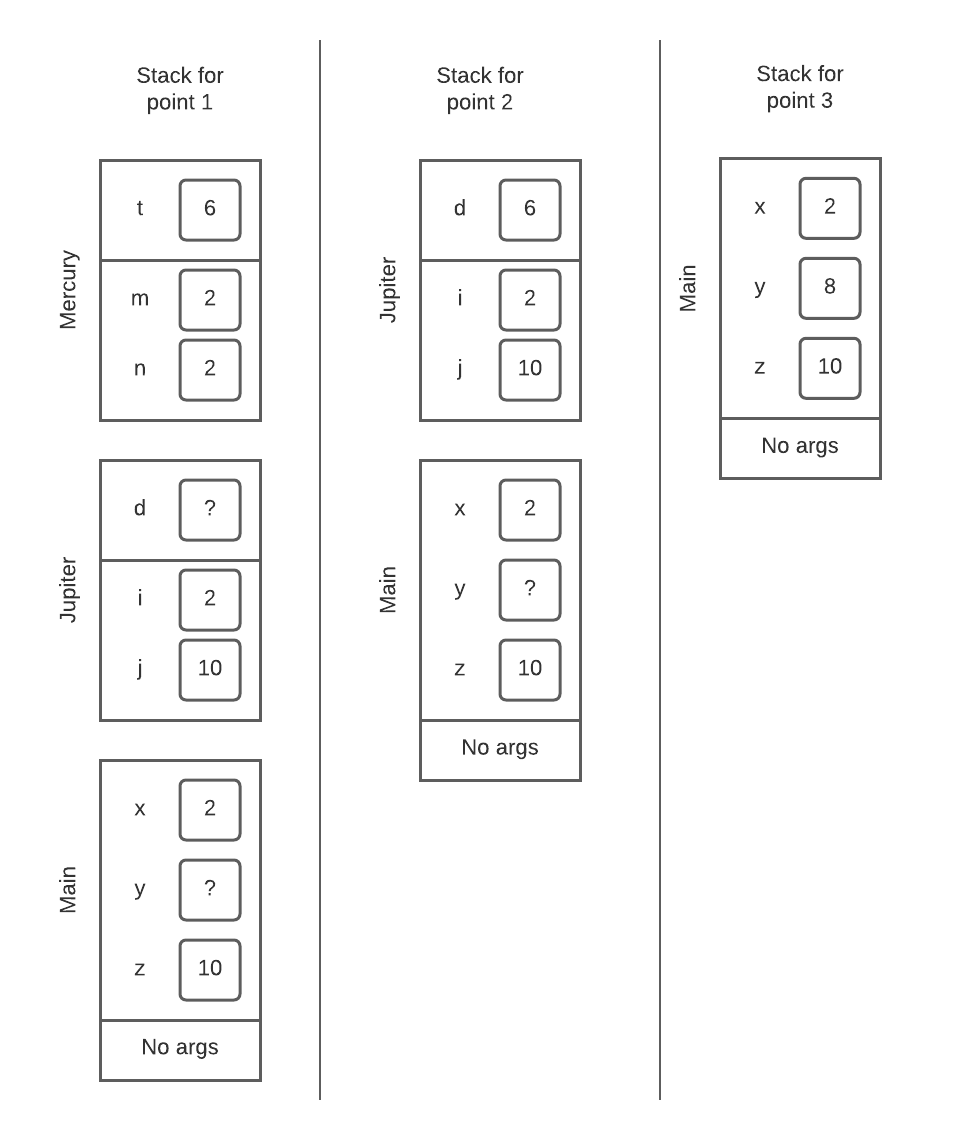
}

Output:

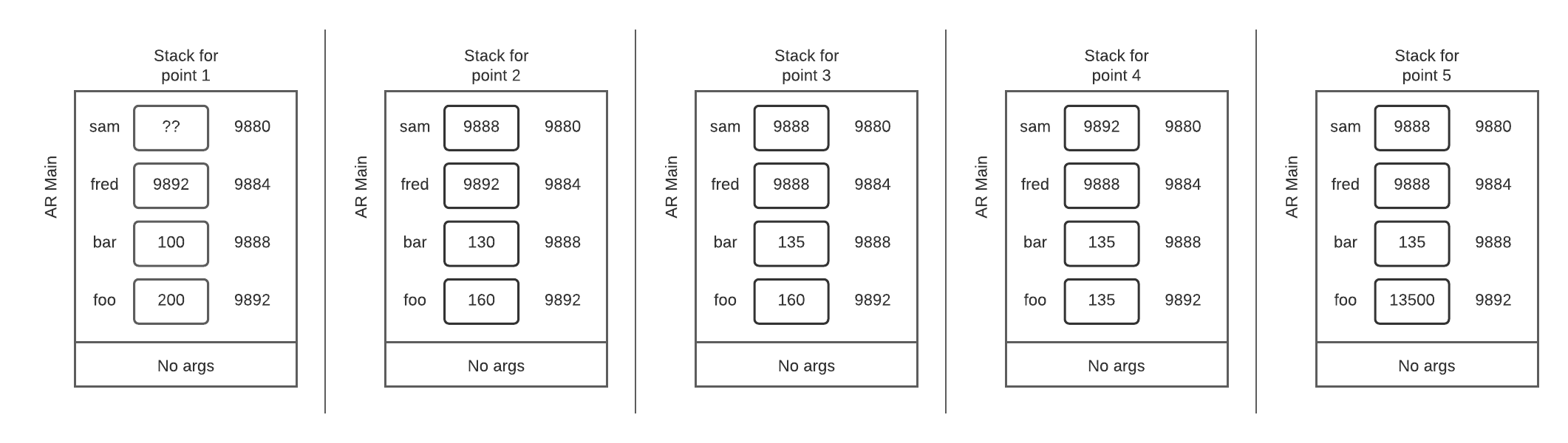
Text

Description automatically generated with medium confidence

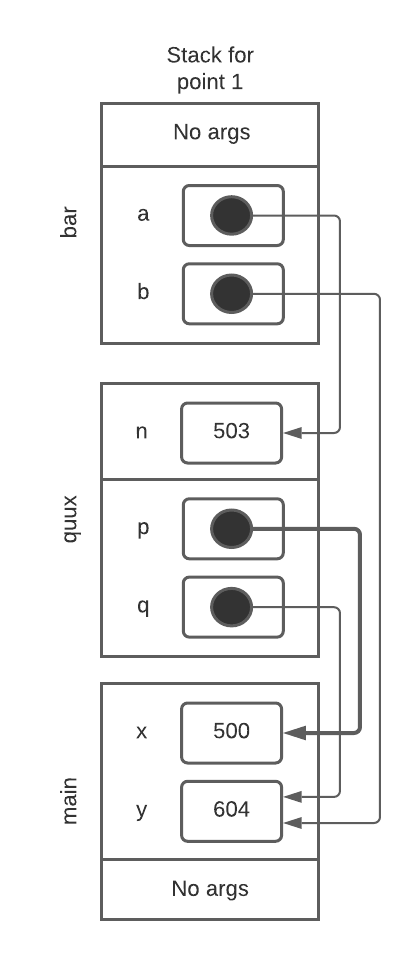
**Exercise B**



**Exercise C**



**Exercise D**



**Exercise E**

Source Code:

/\*

\* File Name: lab2exe\_E.c

\* Assignment: Lab 2 Exercise E

\* Lab section: B04

\* Completed by: Kartik Sharma and Aleksander Berezowski

\* Submission Date: On or before Sept 28, 2021

\*/

#include <stdio.h>

#include <stdlib.h>

void time\_convert(int ms\_time, int \*minutes\_ptr, double \*seconds\_ptr);

int main(void)

{

int millisec;

int minutes;

double seconds;

int nscan;

printf("Enter a time interval as an integer number of milliseconds: ");

nscan = scanf("%d", &millisec);

if (nscan != 1) {

printf("Unable to convert your input to an int.\n");

exit(1);

}

printf("Doing conversion for input of %d ms ... \n", millisec);

time\_convert(millisec, &minutes, &seconds);

printf("That is equivalent to %d minute(s) and %f second(s).\n", minutes,

seconds);

return 0;

}

void time\_convert(int ms\_time, int \*minutes\_ptr, double \*seconds\_ptr)

{

//DESCRIPTION: Converts milliseconds to minutes and seconds

//PROMISES: Update 2 variables with correct values in minutes and seconds via pointers

//REQUIRES: ms\_time must belong to all real numbers, fit within int type, and be in units of milliseconds

//REQUIRES: minutes\_ptr must be a pointer to an address that is an integer variable

//REQUIRES: seconds\_ptr must be a pointer to an address that is a double variable

\*minutes\_ptr = (int)ms\_time/60000;

double temp = (ms\_time/100)%100;

\*seconds\_ptr = temp/10;

}

Output:

Text

Description automatically generated

**Exercise F**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Run # | Your input | What is the value of n | What is the value of i | What is the value of d |
| 1 | 12 0.56 | 2 | 12 | 0.560000 |
| 2 | 5.12 9.65 | 2 | 5 | 0.120000 |
| 3 | 12 ab | 1 | 12 | 1234.500000 |
| 4 | ab 12 | 0 | 333 | 1234.500000 |
| 5 | 5ab 9.56 | 1 | 5 | 1234.500000 |
| 6 | 13 67 | 2 | 13 | 67.000000 |