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...");
```

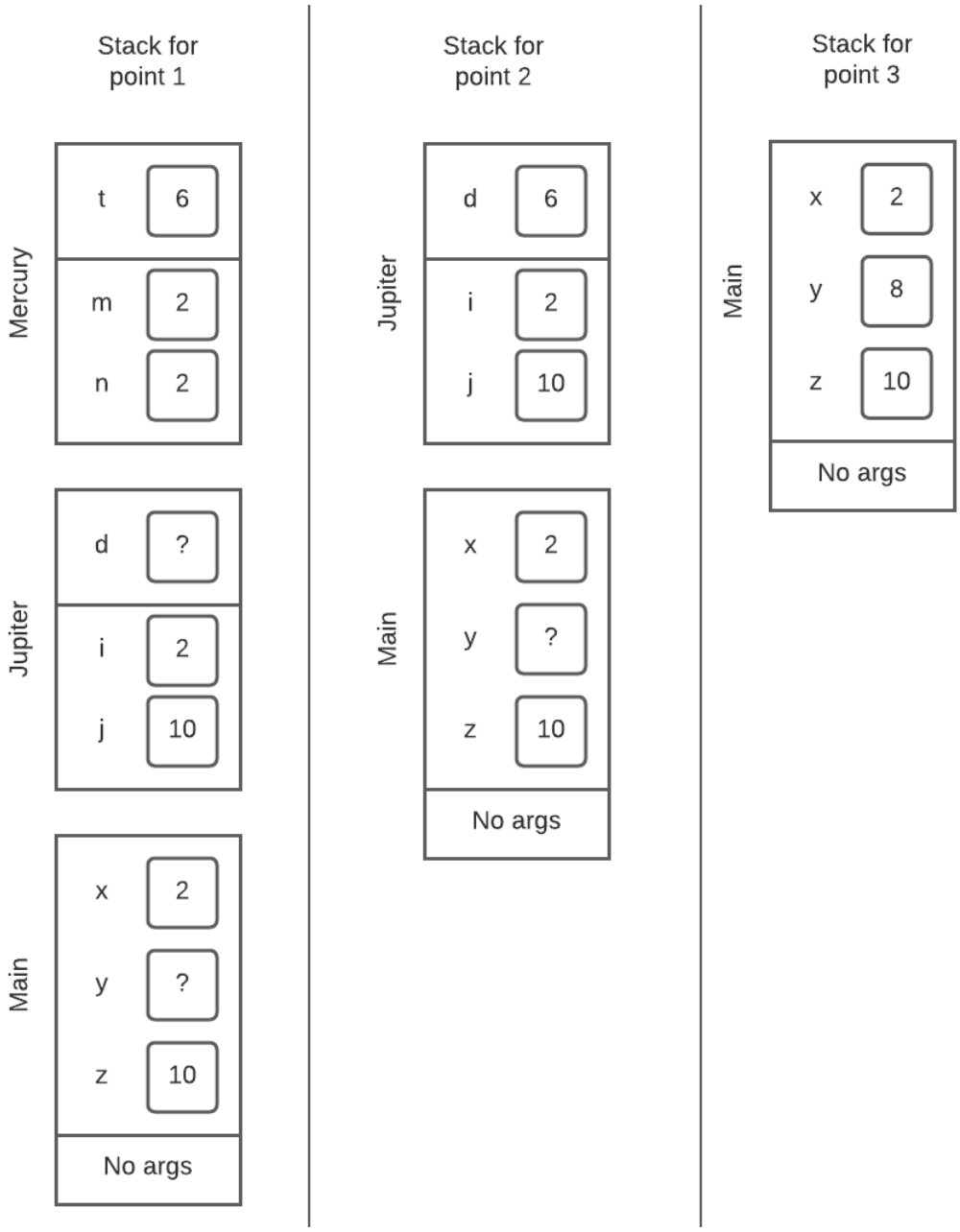
```
        exit(1);
      }
   }

    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\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);
	}
}
```

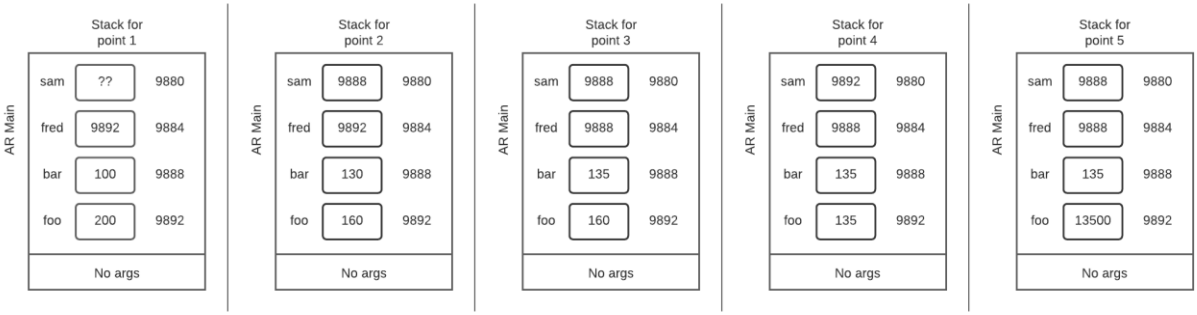
Output:

```
Please enter the velocity at which the projectile is launched (m/sec): 20
Angle           t               d
(deg)           (sec)           (m)
0.000000        0.000000        0.000000
5.000000        0.355738        7.087681
10.000000       0.708768        13.960006
15.000000       1.056404        20.408163
20.000000       1.396001        26.236229
25.000000       1.724972        31.267120
30.000000       2.040816        35.347976
35.000000       2.341128        38.354801
40.000000       2.623623        40.196235
45.000000       2.886150        40.816327
50.000000       3.126712        40.196235
55.000000       3.343478        38.354801
60.000000       3.534798        35.347976
65.000000       3.699215        31.267120
70.000000       3.835480        26.236229
75.000000       3.942554        20.408163
80.000000       4.019623        13.960006
85.000000       4.066101        7.087681
90.000000       4.081633        -0.000000
```

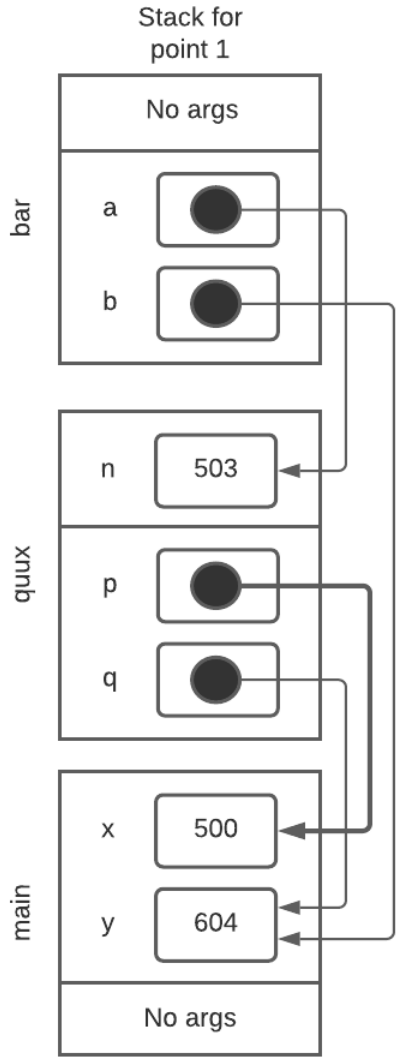
Exercise B


Stack for point 1
Mercury
t 6
m 2
n 2
Jupiter
d ?
i 2
j 10
Main
x 2
y ?
z 10
No args
Stack for point 2
Jupiter
d 6
i 2
j 10
Main
x 2
y ?
z 10
No args
Stack for point 3
Main
x 2
y 8
z 10
No args

Exercise C


Stack for point 1
AR Main
sam ?? 9880
fred 9892 9884
bar 100 9888
foo 200 9892
No args
Stack for point 2
AR Main
sam 9888 9880
fred 9892 9884
bar 130 9888
foo 160 9892
No args
Stack for point 3
AR Main
sam 9888 9880
fred 9888 9884
bar 135 9888
foo 160 9892
No args
Stack for point 4
AR Main
sam 9892 9880
fred 9888 9884
bar 135 9888
foo 135 9892
No args
Stack for point 5
AR Main
sam 9888 9880
fred 9888 9884
bar 135 9888
foo 13500 9892
No args

Exercise D


Stack for
point 1
No args
bar
a
b
n
503
quux
p
q
x
500
main
y
604
No args

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:

```
Enter a time interval as an integer number of milliseconds: 456700
Doing conversion for input of 456700 ms ...
That is equivalent to 7 minute(s) and 6.700000 second(s).
```

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