B.Sc Physics (Major)

Semester I

Physics Core Course 1

Mathematical Physics

Laboratory Notebook

Name : Dhiman Sarkar

Roll Number : 1810093104502

Registration Number : 0091805030584

College Roll Number : 2180051

College : Sukanta Mahavidyalaya

Session : 2018-2019

INDEX

```c
// Finding the Sum and Average of the all Even or Odd numbers from the first given n Natural Numbers

#include<stdio.h>
#include<conio.h>
#include<string.h>

void main()
{
    printf("Find the Sum of all the Even or Odd numbers from the first n Natural Numbers");

    char choice[5];
    int n, sum=0, avg, i, j=0; // n= Number of Natural Numbers, i=General purpose initilizer, j=Number Count

    printf("\n\nPlease enter the last natural number : ");
    scanf("%d",&n);

start: //For restart purpose

    printf("\nWhich type of numbers (Even/Odd) do you want to calculate the sum from first %d Natural numbers? Answer (even/odd) : ",n);
    scanf("%s",choice);

    if(strcmp(choice, "even") == 0) i=2;
    else if(strcmp(choice, "odd") == 0) i=1;
    else {printf("Error!"); goto start;}

    for(i; i<=n; i=i+2)
    {
      j++;
        sum=sum+i;
        avg=sum/j;
    }

    printf("\n\nSum of all the %s numbers from the first %d Natural Numbers is : %d", choice, n, sum);
    printf("\n\nAverage of all the %s numbers (%d numbers) from the first %d Natural Numbers is : %d",choice,j,n,avg);

  printf("\n\n");
    getch();
}

```

Output :

1.

```
## Find the Sum of all the Even or Odd numbers from the first n Natural Numbers ##


Please enter the last natural number : 10


Which type of numbers (Even/Odd) do you want to calculate the sum from first 10
Natural numbers? (even/odd)
Answer : even



Sum of all the even numbers from the first 10 Natural Numbers is : 30


Average of all the even numbers (5 numbers) from the first 10 Natural Numbers is : 6
```

2.

```
## Find the Sum of all the Even or Odd numbers from the first n Natural Numbers ##


Please enter the last natural number : 43


Which type of numbers (Even/Odd) do you want to calculate the sum from first 43
Natural numbers? (even/odd)
Answer : odd



Sum of all the odd numbers from the first 43 Natural Numbers is : 484


Average of all the odd numbers (22 numbers) from the first 43 Natural Numbers is : 22
```

```c
// Find the greatest number and its postition from a given set of numbers

#include<stdio.h>
#include<conio.h>

void main()
{
    printf("## Find the greatest number and its postition from a given set of numbers ##\n\n");

    int n, pos, i;
    printf("How many numbers are there?\n>>");
    scanf("%d",&n);

    float number[n], grtst_num;

    for(i=0; i<n; i++)
    {
        printf("Enter the %th Number >>",i+1);
        scanf("%f",&number[i]);
    }

    // Finding the Greatest number and its Position
    grtst_num = number[0]; //Initilizing comparison with the first number
    pos=1;
    for(i=1; i<n; i++)
    {
        if(number[i]>=grtst_num)
        {
            grtst_num = number[i];
            pos = i+1;
        }
        else ;
    }

    printf("\nThe Greatest number is :: %f and it's at the %dth Position", grtst_num, pos);

    getch();
}
```

Output :

```
## Find the greatest number and its postition from a given set of numbers ##


How many numbers are there?
>>11
Enter the 1th Number >>-54
Enter the 2th Number >>56
Enter the 3th Number >>12
Enter the 4th Number >>-9
Enter the 5th Number >>9
Enter the 6th Number >>15
Enter the 7th Number >>62
Enter the 8th Number >>-8
Enter the 9th Number >>0
Enter the 10th Number >>3
Enter the 11th Number >>4

The Greatest number is :: 62.000000 and it's at the 7th Position
```

```c
// Solving Polynomial Equations using Bisection Method
// Given Function f(x) = x^3 -x^2 -2=0

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<conio.h>


// Function Prototype
float f(float x);
float tolerance(float x1, float x2);
int bisection_method (float x1, float x2, float TOL);

// main() Function
void main()
{
    float TOL, x1, x2;  //TOL = Desiered Tollerence X1,X2=Initial boundary
    printf("####### This Program is to solve a equation by Bisection Method
#######\n\n");
    printf("Please enter Tolerance : ");
    scanf("%f",&TOL);

START: //For Restart purpose

    printf("\nEnter the lower bound of the solution :: ");
  scanf("%f", &x1);
  printf("\nEnter the upper bound of the solution :: ");
  scanf("%f", &x2);

    if (bisection_method(x1,x2,TOL)==0) goto START;
    else ;
    getch();
}


// Defining function f(x)
float f(float x)
{
    float fx = x*x*x -x*x +2;   // Define f(x)
    return fx;
}

//Defining Tolerance function
float tolerance(float x1, float x2)
{
    float TOL = abs(x2-x1); // TOL> Tolerance = x2-x1
    return TOL;
}


//Defining Bisection Method
int bisection_method (float x1, float x2, float TOL)
{
    float x0;
    int n;  //n=number of itteration

    if ( f(x1)*f(x2)>0)
  {
    printf ("\nSolution doesn't exists in the domain (%f,%f)", x1,x2);
```

```
    return (0);

  }
  else ;

    for(n=1; n>=1; n++)
  {
    x0 = (x1+x2)/2;

    if (tolerance(x1,x2)<=TOL)
    {
      printf("\nSolution of the Polynomial equation is :: %f", x0);
            printf("\nNumbner of Iteration :: %d\n\n", n);
      break;
    }
    else if (f(x1)*f(x0)<=0) x2=x0;
    else if (f(x2)*f(x0)<=0) x1=x0;
        else {printf("Error!!");return 0;}
  }
    return 1;
}

```

Output :

```
####### This Program is to solve a equation by Bisection Method #######

Please enter Tolerance : .00001

Enter the lower bound of the solution :: 1

Enter the upper bound of the solution :: 5

Solution doesn't exists in the domain (1.000000,5.000000)



Enter the lower bound of the solution :: -8

Enter the upper bound of the solution :: 5

Solution of the Polynomial equation is :: -1.000001
Number of Iteration :: 22
```

```c
// Solving Polynomial Equation by using Newton-Rhapson Method
// Given Function f(x) = x^3 -x^2 +2

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<conio.h>


// Function Prototype
float F(float x); // Original Function
float f(float x); // Differential Coefficient of F(x)
float tolerance(float x1, float x2);
int NR_method (float x1, float TOL);

// main() Function
void main()
{
    float TOL, x1;  //TOL = Desiered Tollerence X1,X2=Initial boundary
    printf("####### This Program is to solve a equation by Newton-Rhapson Method
#######\n\n");
    printf("Please enter Tolerance : ");
    scanf("%f",&TOL);

    printf("\nEnter the initial approximation of the solution :: ");
  scanf("%f", &x1);

    NR_method(x1,TOL);

    getch();
}


// Defining function f(x) | F(x) stands for anti-derivative i.e. the given function
float F(float x)
{
    float Fx = x*x*x - x*x +2;   // Difine f(x)
    return Fx;
}

// Defining Differential Co-efficiant
float f(float x)
{
    float fx = 3*x*x -4*x;  // Define f'(x)
    return fx;
}

//Defining Tolerance function
float tolerance(float x1, float x2)
{
    float TOL = fabs(x2-x1);
    return TOL;
}


//Defining Bisection Method
int NR_method (float x1, float TOL)
{
    float x2;
    int n;  //n=number of itteration
```

```
60
61      for(n=1; n>0; n++)
62    {
63      x2 = x1 - F(x1)/f(x1);  // p[n] = p[n-1]-f(p[n-1])/f'(p[n-1])
64
65      if (tolerance(x1,x2)<=TOL)
66      {
67        printf("\nSolution of the Polynomial equation is :: %f", x2);
68              printf("\nNumber of Iteration :: %d\n\n", n);
69        break;
70      }
71          else x1=x2;
72    }
73  }
74
```

Output :

```
####### This Program is to solve a equation by Newton-Rhapson Method #######

Please enter Tolerance : .00001

Enter the initial approximation of the solution :: 21

Solution of the Polynomial equation is :: -1.000001
Number of Iteration :: 23
```

```c
// Solving Polynomial Equations using Secant Method
// Given Function f(x) = x^3 -x^2 -2=0

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<conio.h>

#define ERROR .0001


// Function Prototype
float f(float x);
float tolerance(float x1, float x2);
int secant_method (float x1, float x2, float TOL);

// main() Function
void main()
{
    float TOL, x1, x2;  //TOL = Desiered Tollerence X1,X2=Initial boundary
    printf("####### This Program is to solve a equation by Secant Method
#######\n\n");
    printf("Please enter Tolerance : ");
    scanf("%f",&TOL);

START: //For Restart purpose

    printf("\nEnter the lower bound of the solution :: ");
  scanf("%f", &x1);
  printf("\nEnter the upper bound of the solution :: ");
  scanf("%f", &x2);

    if (secant_method(x1,x2,TOL)==0) goto START;
    else ;
    getch();
}


// Defining function f(x)
float f(float x)
{
    float fx = x*x*x -x*x +2;   // Define f(x)
    return fx;
}

//Defining Tolerance function
float tolerance(float x1, float x2)
{
    float TOL = fabs(x2-x1);
    return TOL;
}

//Defining Bisection Method
int secant_method (float x1, float x2, float TOL)
{
    float x3;
    int n;  //n=number of itteration

    if (f(x1)*f(x2)>0)
  {
```

```
    printf ("\nSolution doesn't exists in the domain (%f,%f)\n\n", x1,x2);
    return (0);

  }
  else ;

    for(n=1; n>0; n++)
  {
    x3 = x2 - (f(x2) * (x2-x1) / (f(x2)-f(x1)));

    if (tolerance(x1,x2)<=TOL || fabs(f(x3))<=ERROR)
    {
      printf("\nSolution of the Polynomial equation is :: %f", x3);
            printf("\nNumbner of Iteration :: %d\n\n", n);
      break;
    }
    else if (f(x1)*f(x3)<0) x2=x3;
    else if (f(x2)*f(x3)<0) x1=x3;
        else {printf("Error!!");return 0;}
  }
    return 1;
}

```

Output :

```
####### This Program is to solve a equation by Secant Method #######

Please enter Tolerance : .001

Enter the lower bound of the solution :: -8

Enter the upper bound of the solution :: 9

Solution of the Polynomial equation is :: -0.999981
Numbner of Iteration :: 188
```

```c
// Numerical Integration using Trapezoidal Rule
// Given Function f(x) = 1/(1+x^2)

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>

//Defining the Integral
float f(float x)
{
    float fx= 1/(1+x*x);  // f(x)=1/(1+x^2)
    return fx;
}

void trapezoidal(float xMin, float xMax, float n)
{
    int i; //General purpose initializer
    float h = (xMax-xMin)/n; // h = step size
    float area, marginal_sections_area, middle_sections_area=0;

    for(i=1; i<n-1; i++)
    {
        middle_sections_area = middle_sections_area + f(xMin + i*h);
    }
    marginal_sections_area = f(xMin) + f(xMax);
    area = (h/2)*(marginal_sections_area + 2*middle_sections_area);

    printf("Area between (%f,%f) is : %f",xMin,xMax,area);
}

void main()
{
    printf ("## Numerical Integration using Trapezoidal Rule ##\n\n");

    float xMin, xMax;
    int n;

    printf ("Please enter the lower limit of x : ");
    scanf ("%f", &xMin);
    printf("\nPlease enter the uppper limit of x : ");
    scanf ("%f", &xMax);
    printf ("\nPlease enter the total number of sections :");
    scanf ("%d", &n);
    printf("\n\n");

    trapezoidal(xMin,xMax,n);

    getch();
}
```

Output :

```
## Numerical Integration using Trapezoidal Rule ##

Please enter the lower limit of x : 0

Please enter the uppper limit of x : 1

Please enter the total number of sections :999


Area between (0.000000,1.000000) is : 0.784897
```

```c
// Numerical Integration using Simpson's Rule
// Given Function f(x) = 1/(1+x^2)

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>

//Defining the Integral
float f(float x)
{
    float fx= 1/(1+x*x);  // f(x)=1/(1+x^2)
    return fx;
}

void main()
{
    printf ("## Numerical Integration using Simpson's Rule ##\n\n");

    float xMin, xMax, area;

    printf ("Please enter the lower limit of x : ");
    scanf ("%f", &xMin);
    printf("\nPlease enter the uppper limit of x : ");
    scanf ("%f", &xMax);

    printf("\n\n");

    // Applying Simpson's Rule
    area = ((xMax-xMin)/6) * (f(xMin)+f(xMax)+4*f((xMin-xMax)/2));

    printf("Area of the curve under (%f,%f) is :: %f",xMin,xMax,area);

    getch();
}
```

Output :

```
## Numerical Integration using Simpson's Rule ##

Please enter the lower limit of x : 0

Please enter the uppper limit of x : 1


Area of the curve under (0.000000,1.000000) is :: 0.783333
```

```c
// Finding the Velocity and Accleration from a given set of data of Position and Time by using Forward Difference Numerical Differentiation

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>

// Function to calculate Velocity and Accleration
float state(int n)
{
    int i; // General purpose initializer

    float r[n], t[n], v[n-1], a[n-2];

    printf("     Time            Position\n");
    printf("--------------     --------------\n\n");
    for(i=0; i<=n; i++)
    {
        printf("    t[%d] = ",i+1);
        scanf("%f",&t[i]);
        printf("                    r[%d] = ",i+1);
        scanf("%f",&r[i]);
    }
    for(i=0; i<=n-1; i++)
    {
        v[i] = (r[i+1]-r[i])/(t[i+1]-t[i]);
    }
    for(i=0; i<=n-2; i++)
    {
        a[i] = (v[i+1]-v[i])/(t[i+1]-t[i]);
    }


    printf("   Time              Position             Velocity                    Accleration\n");
    printf("-----------        -------------        ------------         ------------\n\n");
    for(i=0; i<n-2; i++)
    {
        printf("  %f           %f            %f                %f\n",t[i],r[i],v[i],a[i]);
    }
    for(i=n-2; i<n-1; i++)
    {
        printf("  %f           %f            %f                \n",t[i],r[i],v[i]);
    }
    for(i=n-1; i<n; i++)
    {
        printf("  %f           %f                              \n",t[i],r[i],v[i]);
    }
}

//main() Function
void main()
{
    printf("## Finding the Velocity and Accleration from a given set of data of Position and Time by using Forward Difference Numerical Differentiation ##\n\n");
    int n; // n = Number of dataset
```

```
56
57      printf("Please enter the total number of datasets :");
58      scanf("%d",&n);
59      n=n-1; //Counting from 0
60
61      state(n);
62
63      printf("\n\n");
64      getch();
65  }
```

Output :

```
## Finding the Velocity and Accleration from a given set of data of Position and Time by using Forward Difference
Numerical Differentiation ##


Please enter the total number of datasets :10

     Time            Position

--------------     --------------


    t[1] = 0

                    r[1] = 0

    t[2] = 1

                    r[2] = 1

    t[3] = 2

                    r[3] = 4

    t[4] = 3

                    r[4] = 9

    t[5] = 4

                    r[5] = 16

    t[6] = 5

                    r[6] = 25

    t[7] = 6

                    r[7] = 36

    t[8] = 7

                    r[8] = 49

    t[9] = 8

                    r[9] = 64

    t[10] = 9

                    r[10] = 81
   Time             Position             Velocity            Accleration

-----------        -------------        ------------         -------------


  0.000000           0.000000           1.000000             2.000000

  1.000000           1.000000           3.000000             2.000000

  2.000000           4.000000           5.000000             2.000000

  3.000000           9.000000           7.000000             2.000000

  4.000000           16.000000           9.000000             2.000000

  5.000000           25.000000           11.000000             2.000000

  6.000000           36.000000           13.000000             2.000000

  7.000000           49.000000           15.000000

  8.000000           64.000000
```

```c
// Finding the Velocity and Accleration from a given set of data of Position and Time by using Backward Difference Numerical Differentiation

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>

// Function to calculate Velocity and Accleration
float state(int n)
{
    int i; // General purpose initializer

    float r[n], t[n], v[n], a[n];

    n=n-1; //Counting from 0

    printf("     Time            Position\n");
    printf("--------------     --------------\n\n");
    for(i=0; i<=n; i++)
    {
        printf("    t[%d] = ",i+1);
        scanf("%f",&t[i]);
        printf("                    r[%d] = ",i+1);
        scanf("%f",&r[i]);
    }
    //Calculating v[n]
    for(i=n; i>=1; i--)
    {
        v[i] = (r[i-1]-r[i])/(t[i-1]-t[i]);
    }
    //Calculating a[n]
    for(i=n; i>=2; i--)
    {
        a[i] = (v[i-1]-v[i])/(t[i-1]-t[i]);
    }


    printf("   Time              Position             Velocity                Accleration\n");
    printf("-----------        -------------        ------------         ------------\n\n");
    for(i=0; i<1; i++)
    {
        printf("  %f           %f                                \n",t[i],r[i],v[i]);
    }
    for(i=1; i<2; i++)
    {
        printf("  %f           %f            %f                \n",t[i],r[i],v[i]);
    }
    for(i=2; i<n; i++)
    {
        printf("  %f           %f            %f                %f\n",t[i],r[i],v[i],a[i]);
    }
}

//main() Function
void main()
{
```

```
57      printf("## Finding the Velocity and Accleration from a given set of data of
    Position and Time by using Backward Difference Numerical Differentiation ##\n\n");
58
59      int n; // n = Number of dataset
60
61      printf("Please enter the total number of datasets :");
62      scanf("%d",&n);
63
64      state(n);
65
66      printf("\n\n");
67      getch();
68  }
```

Output :

```
## Finding the Velocity and Accleration from a given set of data of Position and Time by using Backward Difference
Numerical Differentiation ##


Please enter the total number of datasets :10

     Time            Position

--------------     --------------


    t[1] = 0

                    r[1] = 0

    t[2] = 1

                    r[2] = 2

    t[3] = 2

                    r[3] = 4

    t[4] = 3

                    r[4] = 9

    t[5] = 4

                    r[5] = 16

    t[6] = 5

                    r[6] = 25

    t[7] = 6

                    r[7] = 36

    t[8] = 7

                    r[8] = 37

    t[9] = 8

                    r[9] = 64

    t[10] = 9

                    r[10] = 81
   Time              Position             Velocity            Accleration

-----------        -------------        ------------         -------------


  0.000000           0.000000

  1.000000           2.000000            2.000000

  2.000000           4.000000            2.000000              -0.000000

  3.000000           9.000000            5.000000              3.000000

  4.000000           16.000000            7.000000              2.000000

  5.000000           25.000000            9.000000              2.000000

  6.000000           36.000000            11.000000              2.000000

  7.000000           37.000000            1.000000              -10.000000

  8.000000           64.000000            27.000000              26.000000
```

```c
// Solving a ODE by using Eular's Method
// Given function f'(x) = 3x^2 +1

#include<stdio.h>
#include<conio.h>

float eular_int_soln(float xi, float yi, float xf, float h);

float dYdX (float x)
{
    float dYdX = 3*x*x +1;
    return dYdX;
}

void main()
{
    printf("## Solving first ordinary differential equation by using Eular's Method ##\n\n");

    float xi, yi, xf, h;

    printf("Please Enter the ititial value conditions\n");
    printf("Initial valu of x >>");
    scanf("%f",&xi);
    printf("Initial value of y at x=%f >>",xi);
    scanf("%f",&yi);
    printf("\nPlease Enter the Final value of x\n>>");
    scanf("%f",&xf);
    printf("Please Enter the step size >>");
    scanf("%f",&h);

    eular_int_soln(xi,yi,xf,h);

    getch();
}

//Defining Eular's Method of solution
float eular_int_soln(float xi, float yi, float xf, float h)
{
    float x, y;

    x=xi;
    y=yi;

    for(x; x<=xf; x=x+h)
    {
        y = y + h*dYdX(x);
    }

    printf("\nSolution for y at x=%f is :: %f",x,y);

    return 0;
}
```

Output :

```
## Solving first ordinary differential equation by using Eular's Method ##


Please Enter the ititial value conditions

Initial valu of x >>0

Initial value of y at x=0.000000 >>3


Please Enter the Final value of x

>>9

Please Enter the step size >>.0001


Solution for y at x=9.000046 is :: 740.148132
```