Assignment Answer Sheet

SEMESTER-____3rd____

<mark>MATH HONS</mark>   /GE   /PURE GENERAL   (Please tick the correct option)

Name of student: _BIKRAM BAKSHI_____

   College Roll Number:_1915033_____

   Registration Number:_1061911400143_____

   Department:_MATHEMATICS____(Fore GE Students/Hons Student)

Course Code:_MTMACOR07P_____   Course Type: _CORE_____

Course Name/Title: _NUMERICAL METHODS LAB_____

Res. Address: VIVEKANANDA PALLY, TAKI ROAD, BARASAT, NORTH 24 PGS., KOL-700124

 Mobile Number: _7998872014_____

DEPARTMENT OF MATHEMATICS

BARASAT GOVERNMENT

COLLEGE 10, K.N.C ROAD,

BARASAT KOLKATA-700124

2020-21

1.Calculate the sum 1/1 + 1/2 + 1/3 + 1/4 + …. + 1/N.

```c
/*
 * C Program to find the Sum of Series 1 + 1/2 + 1/3 + 1/4 + ... + 1/N
 */
#include <stdio.h>
int main()
{
    double number, sum = 0, i;
    printf("\n enter the number ");
    scanf("%lf", &number);
    for (i = 1; i <= number; i++)
    {
        sum = sum + (1 / i);
        if (i == 1)
            printf("\n 1 +");
        else if (i == number)
            printf(" (1 / %lf)", i);
        else
            printf(" (1 / %lf) + ", i);
    }
    printf("\n The sum of the given series is %.2lf\n", sum);
    return 0;
}
```

OUTPUT:

```
suman@hp-ubuntu:~/Working Stuff/Proj-Assign-Practice/Bikram_C$ gcc Q01.c && ./a.out

 enter the number 8

 1 + (1 / 2.000000) +  (1 / 3.000000) +  (1 / 4.000000) +  (1 / 5.000000) +  (1 / 6.
000000) +  (1 / 7.000000) +  (1 / 8.000000)
 The sum of the given series is 2.72
suman@hp-ubuntu:~/Working Stuff/Proj-Assign-Practice/Bikram_C$ 
```

2.Enter 100 integers into an array and sort them in an ascending order.

```c
//C program to accept N numbers and arrange them in an ascending order
#include <stdio.h>
int main()
{
    int i, j, a, n, number[30];
    printf("Enter the value of N \n");
    scanf("%d", &n);
    printf("Enter the numbers \n");
    for (i = 0; i < n; ++i)
        scanf("%d", &number[i]);
    for (i = 0; i < n; ++i)
    {
        for (j = i + 1; j < n; ++j)
        {
            if (number[i] > number[j])
            {
                a =  number[i];
                number[i] = number[j];
                number[j] = a;
            }

        }
    }
    printf("The numbers arranged in ascending order are given below \n");
    for (i = 0; i < n; ++i)
        printf("%d\n", number[i]);

}
```

OUTPUT:

```
Enter the value of N
6
Enter the numbers
32
65
96
45
456
35
The numbers arranged in ascending order are given below
32
35
45
65
96
456
```

3. Program for Bisection Method in C

```c
// A program of bisection method in c

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
float f(float x)
{
        float sum;
        sum = pow(x, 3) + x + 3;
        return sum;
}
int main()
{
        float a, b, e, m;
        printf("This Program Illustrates the bisection method in C:\n");
        printf("x^3 + 3*x - 5 = 0\n");
        step:
        printf("Enter the Value of a and b: ");
        scanf("%f%f",&a, &b);
        printf("Enter tolerable Error: \n");
        scanf("%f",&e);
        if (f(a)*f(b) > 0)
        {
                goto step;
        }
        step1:
        m = (a+b)/2;
        if (f(m) == 0){
                printf("the root is : %f", m);
                exit(0);
        }
        else if (f(a) * f(m) < 0)
                b = m;
        else if (f(b) * f(m) < 0)
                a = m;
        if (fabs (a - b) > e){
                goto step1;
        }
        printf("the root is : %f\n", m);
        return 0;
}
```

OUTPUT:

```
This Program Illustrates the bisection method in C:
x^3 + 3*x - 5 = 0
Enter the Value of a and b: -1 -2
Enter tolerable Error:
0.0004
the root is : -1.213623
```

4. Write a C program and run to find the smallest positive root of the equation $tanx + a\ sin^2x=1, a=.3$ By Newton Raphson Method correct upto 5 significant figure.

```c
/* Solution by Newton Raphson Method */

#include <stdio.h>
#include <math.h>
#define err 0.00001
#define ITNO 20
#define F(x)  tan(x)+a* pow(sin(x),2)-1
#define FD(x) (1/pow(cos(x),2))+a*sin(2*x)
int main()
{
  int count;
  float x0,xn,fx,fdx,a=3;
  printf("Give the initial approximation\n");
  scanf("%f",&x0);
  count=1;
  begin:
  fx=F(x0);
  fdx= FD(x0);
    xn=x0-(fx/fdx);
    printf("\n n=%d xn=%f",count,xn);
    if(fabs(xn-x0)<err)
     printf("\n the root is %f",xn);
    else
    {
     x0=xn;
     count=count+1;
     if(count<=ITNO)
     {
       goto begin;
     }
     else
     {
       printf("the solution does not converge");
     }
```

```
    }
}
```

OUTPUT:

```
Give the initial approximation
0.1

 n=1 xn=0.641547
 n=2 xn=0.456340
 n=3 xn=0.436017
 n=4 xn=0.435727
 n=5 xn=0.435727
 the root is 0.435727
```

5. Code for JACOBIAN METHOD in C Programming

```c
#include<stdio.h>
#include<math.h>
#define ESP 0.0001
#define X1(x2,x3) ((17 - 20*(x2) + 2*(x3))/20)
#define X2(x1,x3) ((-18 - 3*(x1) + (x3))/20)
#define X3(x1,x2) ((25 - 2*(x1) + 3*(x2))/20)
int main()
{
        double x1=0,x2=0,x3=0,y1,y2,y3;
        int i=0;
        printf("\n _____\n");
        printf("\n x1\t\t  x2\t\t x3\n");
        printf("\n _____\n");
        printf("\n %f\t %f\t %f",x1,x2,x3);
        do
        {
                y1=X1(x2,x3);
                y2=X2(x1,x3);
                y3=X3(x1,x2);
                if (fabs(y1-x1)<ESP&&fabs(y2-x2)<ESP&&fabs(y3-x3)<ESP)
                {
                        printf("\n _____\n");
                        printf("\n\nx1=%.3lf",y1);
                        printf("\n\nx2=%.3lf",y2);
                        printf("\n\nx3=%.3lf\n",y3);
                        i=1;
                }
                else
                {
                        x1=y1;
                        x2=y2;

                        x3=y3;
                        printf("\n %f\t\t%f\t\t%f\n",x1,x2,x3);
                }
```

```
    }
    while(i!=1);
}
```

OUTPUT:

```
_____

x1              x2              x3

_____

0.000000      0.000000      0.000000
0.850000          -0.900000          1.250000

1.875000          -0.965000          1.030000

1.918000          -1.129750          0.917750

2.071525          -1.141812          0.888738

2.080686          -1.166292          0.871576

2.103449          -1.168524          0.866988

2.105223          -1.172168          0.864376

2.108606          -1.172565          0.863653

2.108930          -1.173108          0.863255

2.109434          -1.173177          0.863141

_____

x1=2.109

x2=-1.173

x3=0.863
```

6. Source Code for Lagrange Interpolation in C:

```c
#include<stdio.h>
int main()
{
        float x[100],y[100],a,s=1,t=1,k=0;
        int n,i,j;
        printf("\n\n Enter the number of the terms of the table: ");
        scanf("%d",&n);
        printf("\n\n Enter the respective values of the variables x and y: \n");
        for(i=0; i<n; i++)
        {
                scanf ("%f",&x[i]);
                scanf("%f",&y[i]);
                }
                printf("\n\n The table you entered is as follows :\n\n");
                for(i=0; i<n; i++)
                {
                        printf("%0.3f\t%0.3f",x[i],y[i]);
                        printf("\n");
                        }
k=0;
printf("\n Enter the Value of the x to find respective value of y\n");
scanf("%f",&a);
for(i=0;i<n;i++)
{
        s=1;
        t=1;
        for (j=0;j<n;j++)
        {
                if(j!=i)
                {
                        s=s*(a-x[j]);
                        t=t*(x[i]-x[j]);
                }
        }
        k=k+((s/t)*y[i]);
```

}
printf("\n\n THe respective value of the variable y is :%f\n",k); }

OUTPUT:

```
Enter the number of the terms of the table: 4

 Enter the respective values of the variables x and y:
0
8
1
10
4
75
5
120


 The table you entered is as follows :

0.000    8.000
1.000    10.000
4.000    75.000
5.000    120.000

 Enter the Value of the x to find respective value of y
4


 THe respective value of the variable y is :75.000000
```

7. Write a C program and run to Evaluate $\int_{4}^{7} \frac{\log(1+ax+a^2)}{a+x}\,dx$, A=.8 by Simson's Rule correct upto 5 significant figures.

```c
/*Simson*/
#include <stdio.h>
#include <math.h>
int main( )
{
  int n,i;
  float a,b,h,sum1,sum2,sum3,intvle;
  float F(float x);
  printf(" give the initial value of limit a\n");
  scanf("%f",&a);
  printf( "give the value of final limit b\n");
  scanf("%f",&b);
  printf( "give the value of interval 'n'\n");
  scanf("%d",&n);
  h=(b-a)/n;
  sum1=(F(a)+F(b));
  sum2=0;
  for (i=1;i<n;i=i+2)
  {
   sum2=sum2+F(a+i*h);
  }
  sum3=0;
  for (i=2;i<n-1;i=i+2)
  {
   sum3=sum3+F(a+i*h);
  }
  intvle=h*(sum1+4*sum2+2*sum3)/3;
  printf("\n");
  printf("a..........b............n............Value of Integration\n\n\n");
  printf( "\n%f  %f    %d          %5f\n",a,b,n,intvle);
}
float F(float x)
  {
```

```
  float f;
  f=(log(1+.8*x+.64))/(.8+x);
  return (f);
}
```

OUTPUT:

```
 give the initial value of limit a
1
give the value of final limit b
6
give the value of interval 'n'
4

a..........b............n............Value of Integration


1.000000  6.000000    4            1.793256
```

8. Power Method for Dominant Eigen Values

```c
#include<stdio.h>
#include<math.h>
int main()
{
   int i,j,n;
   float A[40][40],x[40],z[40],e[40],zmax,emax;
   printf("\nEnter the order of matrix:");
   scanf("%d",&n);
   printf("\nEnter matrix elements row-wise\n");
   for(i=1; i<=n; i++)
   {
      for(j=1; j<=n; j++)
      {
         printf("A[%d][%d]=", i,j);
         scanf("%f",&A[i][j]);
      }
   }
   printf("\nEnter the column vector\n");
   for(i=1; i<=n; i++)
   {
      printf("X[%d]=",i);
      scanf("%f",&x[i]);
      }
   do
   {
      for(i=1; i<=n; i++)
      {
         z[i]=0;
         for(j=1; j<=n; j++)
         {
          z[i]=z[i]+A[i][j]*x[j];
         }
      }
      zmax=fabs(z[1]);
      for(i=2; i<=n; i++)
      {
```

```
        if((fabs(z[i]))>zmax)
              zmax=fabs(z[i]);
      }
      for(i=1; i<=n; i++)
      {
        z[i]=z[i]/zmax;
      }
      for(i=1; i<=n; i++)
      {
        e[i]=0;
        e[i]=fabs((fabs(z[i]))-(fabs(x[i])));
      }
      emax=e[1];
      for(i=2; i<=n; i++)
      {
        if(e[i]>emax)
        emax=e[i];
      }
      for(i=1; i<=n; i++)
      {
        x[i]=z[i];
      }
    }
    while(emax>0.001);
    printf("\n The required eigen value is %f",zmax);
    printf("\n\nThe required eigen vector is :%f\n",emax);

    for(i=1; i<=n; i++)
    {
        printf("%f\t",z[i]);

    }
}
```

OUTPUT:

```
Enter the order of matrix:3

Enter matrix elements row-wise
A[1][1]=2
A[1][2]=-1
A[1][3]=0
A[2][1]=-1
A[2][2]=2
A[2][3]=-1
A[3][1]=0
A[3][2]=-1
A[3][3]=2

Enter the column vector
X[1]=1
X[2]=0
X[3]=0

 The required eigen value is 3.414214

The required eigen vector is :0.000956
0.708459        -1.000000 _     0.705754
```
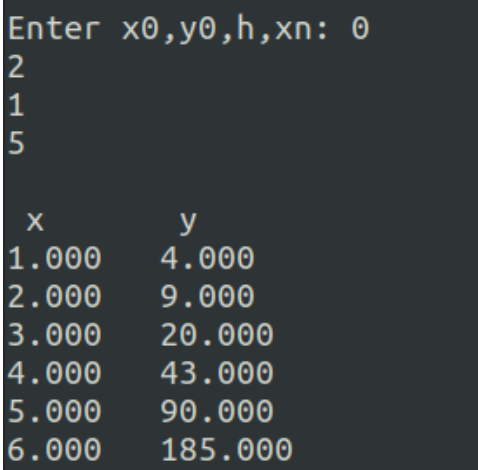
9. Euler's method in C to solve the ordinary differential equation dy/dx = x+y.

```
#include<stdio.h>
float fun(float x,float y)
{
   float f;
   f=x+y;
   return f;
}
int main()
{
   float a,b,x,y,h,t,k;
   printf("\nEnter x0,y0,h,xn: ");
   scanf("%f%f%f%f",&a,&b,&h,&t);
   x=a;
   y=b;
   printf("\n x\t y\n");
   while(x<=t)
   {
      k=h*fun(x,y);
      y=y+k;
      x=x+h;
      printf("%0.3f\t%0.3f\n",x,y);
   }
}
```

OUTPUT:

```
Enter x0,y0,h,xn: 0
2
1
5

 x          y
1.000    4.000
2.000    9.000
3.000    20.000
4.000    43.000
5.000    90.000
6.000    185.000
```

10. Write a C program and run to find the value of y at x=.4 from the ODE $\frac{dy}{dx} = \frac{x-y}{x+y}$ given that y(0)=1 by Runge Kutta Method correct uoto 3 significant figures.

```c
/*Runge kutta*/
#include <stdio.h>
#include <math.h>
float f(float x,float y);
int main( )
{
  float x0,y0,m1,m2,m3,m4,m,x,y,h,xn;
  printf("Enter x0,y0,xn,h :\n");
  scanf("%f%f%f%f",&x0,&y0,&xn,&h);
  x=x0;
  y=y0;
  printf("\n\nX\t\tY\n");
  while(x<xn)
  {
   m1=f(x0,y0);
   m2=f((x0+h/2.0),(y0+m1*h/2.0));
   m3=f((x0+h/2.0),(y0+m2*h/2.0));
   m4=f((x0+h),(y0+m3*h));
   m= ((m1+2*m2+2*m3+m4)/6);
  y=y+m*h;
  x=x+h;
  }
  printf("%f\t%f\n",x,y);
}
float f(float x, float y)
{
  float m;
  m=(x-y)/(x+y);
  return m;
}
```

OUTPUT:

```
Enter x0,y0,xn,h :
0
2
1
5

X                Y
5.000000         3.833394
```