OUTPUT

```
Wellcome to the Main Menu


1.     Sum of the candidate number digits
2.     Calculate the roots of ax2+bx+c=0
3.     Calculate Structure a(x+1)n-1
4.     Find numbers and sum of all numbers between 200 and 300
5.     Display the product of the data in the array
6.     Find the maximum number using the pointers
7.     Process the data file
8.     Exit Program
Select from options (1,2,3,4,5,6,7,8) : 1

Please Enter Candidate Number = 12345
Cube of (sum-1)/2 = 343.000000
Wellcome to the Main Menu


1.     Sum of the candidate number digits
2.     Calculate the roots of ax2+bx+c=0
3.     Calculate Structure a(x+1)n-1
4.     Find numbers and sum of all numbers between 200 and 300
5.     Display the product of the data in the array
6.     Find the maximum number using the pointers
7.     Process the data file
8.     Exit Program
Select from options (1,2,3,4,5,6,7,8) : 2

Please Enter  a =       12

Please Enter  b =       34

Please Enter  c =       5

Quadratic Roots are real and distinct Root 1 = -0.155605
Root 2 = -2.677729

Wellcome to the Main Menu


1.     Sum of the candidate number digits
2.     Calculate the roots of ax2+bx+c=0
3.     Calculate Structure a(x+1)n-1
4.     Find numbers and sum of all numbers between 200 and 300
5.     Display the product of the data in the array
6.     Find the maximum number using the pointers
7.     Process the data file
8.     Exit Program
Select from options (1,2,3,4,5,6,7,8) : 3
```

```
2.     Calculate the roots of ax2+bx+c=0
3.     Calculate Structure a(x+1)n-1
4.     Find numbers and sum of all numbers between 200 and 300
5.     Display the product of the data in the array
6.     Find the maximum number using the pointers
7.     Process the data file
8.     Exit Program
Select from options (1,2,3,4,5,6,7,8) : 3

Please enter the real # 1 =12

Please enter the real # 2 =4

Please enter the integer number:5

Calculated structure of a(x+1)n-1 is given as follow :
12(4+1)4 = 7500
12(4+1)3 = 1500
12(4+1)2 = 300
12(4+1)1 = 60
12(4+1)0 = 12
Wellcome to the Main Menu


1.     Sum of the candidate number digits
2.     Calculate the roots of ax2+bx+c=0
3.     Calculate Structure a(x+1)n-1
4.     Find numbers and sum of all numbers between 200 and 300
5.     Display the product of the data in the array
6.     Find the maximum number using the pointers
7.     Process the data file
8.     Exit Program
Select from options (1,2,3,4,5,6,7,8) : 4

Please enter the divisor = 23

The numbers between 200 and 300 that are divisible by 23 are as follow : 207     230     253     276     299
The sum of the numbers are 1265
Wellcome to the Main Menu
```

```
1.     Sum of the candidate number digits
2.     Calculate the roots of ax2+bx+c=0
3.     Calculate Structure a(x+1)n-1
4.     Find numbers and sum of all numbers between 200 and 300
5.     Display the product of the data in the array
6.     Find the maximum number using the pointers
7.     Process the data file
8.     Exit Program
Select from options (1,2,3,4,5,6,7,8) : 5

Enter 6 values as a data to the array = 16
32
11
22
40
2017

Please enter the new data for the last value of the array23

The product of the data is = 113991680
Wellcome to the Main Menu


1.     Sum of the candidate number digits
2.     Calculate the roots of ax2+bx+c=0
3.     Calculate Structure a(x+1)n-1
4.     Find numbers and sum of all numbers between 200 and 300
5.     Display the product of the data in the array
6.     Find the maximum number using the pointers
7.     Process the data file
8.     Exit Program
Select from options (1,2,3,4,5,6,7,8) : 6

Please enter 3 numbers =23
56
1

Maximum number = 56
```

```
Wellcome to the Main Menu


1.     Sum of the candidate number digits
2.     Calculate the roots of ax2+bx+c=0
3.     Calculate Structure a(x+1)n-1
4.     Find numbers and sum of all numbers between 200 and 300
5.     Display the product of the data in the array
6.     Find the maximum number using the pointers
7.     Process the data file
8.     Exit Program
Select from options (1,2,3,4,5,6,7,8) : 7

a). Print out the values
b). Find the maximum value
c). Calculate the RMS (Root Means Square)
d). Count the number of Negative values
e). Exit back to main menu
Please select option (a,b,c,d,e) =      a

The values from the files are =
1       6       4       5       5       9       12      14      15      -17     -19     21      -23     0       37
 0       -31     32      34      -37     -39
a). Print out the values
b). Find the maximum value
c). Calculate the RMS (Root Means Square)
d). Count the number of Negative values
e). Exit back to main menu
Please select option (a,b,c,d,e) =      b

The maximum value is 37
a). Print out the values
b). Find the maximum value
c). Calculate the RMS (Root Means Square)
d). Count the number of Negative values
e). Exit back to main menu
Please select option (a,b,c,d,e) =      c

The RMS (Root Mean Square ) is 21.563858
a). Print out the values
b). Find the maximum value
c). Calculate the RMS (Root Means Square)
d). Count the number of Negative values
e). Exit back to main menu
Please select option (a,b,c,d,e) =      d

The count of the negative numbers is 6
a). Print out the values
b). Find the maximum value
c). Calculate the RMS (Root Means Square)
```

```
d). Count the number of Negative values
e). Exit back to main menu
Please select option (a,b,c,d,e) =      c

The RMS (Root Mean Square ) is 21.563858
a). Print out the values
b). Find the maximum value
c). Calculate the RMS (Root Means Square)
d). Count the number of Negative values
e). Exit back to main menu
Please select option (a,b,c,d,e) =      d

The count of the negative numbers is 6
a). Print out the values
b). Find the maximum value
c). Calculate the RMS (Root Means Square)
d). Count the number of Negative values
e). Exit back to main menu
Please select option (a,b,c,d,e) =      e

Wellcome to the Main Menu


1.      Sum of the candidate number digits
2.      Calculate the roots of ax2+bx+c=0
3.      Calculate Structure a(x+1)n-1
4.      Find numbers and sum of all numbers between 200 and 300
5.      Display the product of the data in the array
6.      Find the maximum number using the pointers
7.      Process the data file
8.      Exit Program
Select from options (1,2,3,4,5,6,7,8) : 8

Good Bye
--------------------------------
Process exited after 59.65 seconds with return value 0
Press any key to continue . . .
```

SOURCE CODE

```
//C Program assignment
//Written By:     Candidate Number :  181521


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



//function that will calculate the roots of the quadratic equations
void Quadratic(int a, int b, int c)
{
        //required variables
        float value;
        float root1, root2;
   float real_num, imag_num;


        if (a == 0 )
        {
                printf("Error: Roots cannot be determined \n");
        }
        else if(b == 0 )
        {
                printf("Error: Roots cannot be determined \n");
        }
        else if( c == 0)
  {
     printf("Error: Roots cannot be determined \n");
```

Writingsol

```
}
else
{
        //calculate the b2-4ac
   value = pow(b,2) - (4.0 * a * c);
   //if b2-4ac>0 then the value is real and equal
   //else if b2-4ac=0 then the value is real and distinct
   //else will be imaginary
                    if (value > 0)
   {
        printf("\nQuadratic Roots are real and distinct ");
      //calculate the 1st root
                              root1 =(-b + sqrt(value)) / (2.0 * a);
                              //calculate the 2nd root
      root2 =(-b - sqrt(value)) / (2.0 * a);
      //show both the roots to screen
      printf("Root 1 = %f \n", root1);
      printf("Root 2 = %f  \n", root2);
   }
   else if (value == 0)
   {
      printf("\nQuadratic Roots are real and equal");
      //calculate 1st root
      root1 = -b / (2.0 * a);
      //1st root will be the 2nd root as well
      root2 = root1;
      //show both the roots to screen
      printf("Root 1 = %f\n", root1);
      printf("Root 2 = %f\n", root2);
```

Writingsol

```
    }
    else if (value < 0 )
    {
       printf("\nnQuadratic Roots are imaginary ");
       //calculate the 1st root
       real_num = -b / (2.0 * a) ;
       //calculate the 2nd root
       imag_num = sqrt(abs(value)) / (2.0 * a);
       //show the real and imaginary part
       printf("Root1 = %f +i %f\n", real_num, imag_num);
       printf("Root2 = %f  -i %f\n", real_num, imag_num);
    }
  }
}

//function to read data from file and return total number of data available
int ReadFile(int filearray[50])
{
        //counter
        int counter=0;
        //file pointer
        FILE *file;
        if (file = fopen("data.txt", "r"))
        {
    while (fscanf(file, "%d", &filearray[counter]) != EOF)
                {
       ++counter;
    }
    fclose(file);
```

Writingsol

```
    }
    else
    {
            counter=-1;
            }
    return counter;
}

//function to display the array data
void ShowArray(int filearray[50],int count)
{
            int i=0;
            printf("\nThe values from the files are =\n");
            while(i<count)
            {
                        printf("%d\t ",filearray[i]);
                        i++;
            }
}

//function to find the maximum number
int FindMaximum(int filearray[50],int count)
{
            //variable to hold the maximum number
            int max;
            int i=0;
            max=filearray[0];
            while(i<count)
            {
```

Writingsol

```
		if(filearray[i]>max)
			max=filearray[i];
		i++;
	}
	return max;
}

//function to calculate the number of negative valules
int CountNegative(int filearray[50],int count)
{
	//variable to hold the count of the negative numbers
	int c=0;
	int i=0;
	while(i<count)
	{
		if(filearray[i]<0)
			c++;
		i++;
	}
	return c;
}

//function to calculate the RMS
float CalculateRMS(int filearray[50],int count)
{
	//required variable
	int sum=0;
	float avg=0;
	float sq_avg=0;
```

Writingsol

```
	int i=0;
	while(i<count)
	{
		sum=sum+(pow(filearray[i],2));
		i++;
	}
	return sqrt(sum/count);
}

int main(void)
{
	//Required Variables
	int option;		//choice of menu from user
	char option2;	//choice of menu 2 from user
	int student_num;		//student candidate number
	float square,cube;		//temporary variable
	int num;		//digits variable
	int sum=0;		//sum of the candidate numbers
	int a;	//a root of quadratic equations
	int b;	//b root of quadratic equations
	int c;	//c roots of the quadratic equations
	int x;	// x for a(x+1)n-1
	int n;	// n for a(x+1)n-1
	int result;		//result of a(x+1)n-1
	int i;	//counter variable for iterators
	int divisor;		//to hold the divisor entered by the user
	int data[6];		//to store the array
	int *pointer;		//pointer
	int max;		//maximum
```

Writingsol

```
int total=0;        //total numbers
float rms;          //rms
int filearray[50];          //data of file

//iterator to execute the menu atleast once
do
{
        //show the main menu
        printf("\nWellcome to the Main Menu\n\n");
        printf("\n1.    Sum of the candidate number digits");
        printf("\n2.    Calculate the roots of ax2+bx+c=0");
        printf("\n3.    Calculate Structure a(x+1)n-1");
        printf("\n4.    Find numbers and sum of all numbers between 200 and 300");
        printf("\n5.    Display the product of the data in the array ");
        printf("\n6.    Find the maximum number using the pointers");
        printf("\n7.    Process the data file");
        printf("\n8.    Exit Program");

        //ask for the main menu choice
        printf("\nSelect from options (1,2,3,4,5,6,7,8) :\t");
        scanf("%d",&option);

        sum = 0;

        //selection according to the choice of user
        if(option==1)
        {
                //ask for the student candidate number
                printf("\nPlease Enter Candidate Number =\t");
```

Writingsol

```
        scanf("%d",&student_num);
        //iterator unless num is divisible by 10
        while (student_num > 0)
        {
           sum  = sum + (student_num % 10);
           student_num /= 10;
        }

        //validate if the sum is even number
        if(sum%2==0)
        {
                //calculate square of the (sum/2)
                square=pow((float)(sum/2),2);
                printf("Square of (sum/2) = %f",square);
        }
        else
        {
                //calculate cube of the (sum-1)/2
                cube=pow((float)(sum/2),3);
                printf("Cube of (sum-1)/2 = %f",cube);
        }
}
else if(option==2)
{
        //ask user for 2 input values for a,b,c to calculate quadratic equations
        printf("\nPlease Enter  a =\t");
        scanf("%d",&a);
        printf("\nPlease Enter b =\t");
        scanf("%d",&b);
```

Writingsol

```
            printf("\nPlease Enter c =\t");
            scanf("%d",&c);
            //call quadratic function
            Quadratic(a,b,c);
}
else if(option==3)
{
            //ask for  real  number
            printf("\nPlease enter the real # 1 =");
            scanf("%d",&a);
            printf("\nPlease enter the real # 2 =");
            scanf("%d",&x);
            printf("\nPlease enter the integer number:");
            scanf("%d",&n);
            //initialize the counter
            i=n;
            //show the descriptive text
            printf("\nCalculated structure of a(x+1)n-1 is given as follow :");
            //iteration to calculate structure a(x+1)n-1
            while(i>0)
            {
                        //display the structure
                        result = a * (pow((x+1),(i-1)));
                        printf("\n%d(%d+1)%d = %d",a,x,i-1,result);
                        i--;
            }
}
else if(option==4)
{
```

Writingsol

```
                    //prompt for the number that is used to be divisor
                    printf("\nPlease enter the divisor = ");
                    scanf("%d",&divisor);

                    //initial number for the counter
                    i=200;
                    //show the descriptive message
                    printf("\nThe numbers between 200 and 300 that are divisible by %d are as
follow : ",divisor);
                    //iterator to find and calculate the sum of the numbers divisible by the divisor
number
                    while(i<=300)
                    {
                            //validate if the number is divisible
                            if(i%divisor==0)
                            {
                                    //take the summation of the number
                                    sum+=i;
                                    //display the values
                                    printf("%d\t ",i);
                            }
                            i++;
                    }
                    //show the sum of the divisible numbers
                    printf("\nThe sum of the numbers are %d",sum);
            }
            else if(option==5)
            {
                    //display discriptive message
```

Writingsol

```
            printf("\nEnter 6 values as a data to the array =\t");
            //iterator to receive input for all data in the array
            i=0;
            while(i<6)
            {
                  //receive the input from user
                  scanf("%d",&data[i]);
                  i++;
            }

            result=1;
            //ask the user to replace the last value
            printf("\nPlease enter the new data for the last value of the array");
            scanf("%d",&data[5]);
            //calculate the product

            i=0;
            while(i<6)
            {
                  result = result*data[i];
                  i++;
            }
            //show the data
            printf("\nThe product of the data is = %d",result);
}
else if(option==6)
{
            pointer=(int *) malloc(3*2);
```

Writingsol

```
            //ask for three values
            printf("\nPlease enter 3 numbers =");
            scanf("%d",pointer);
            scanf("%d",(pointer+1));
            scanf("%d",(pointer+2));

            max = *pointer;
            i=1;
            while(i<3)
            {
                  if(*(pointer+i)>max)
                        max=*(pointer+i);
                  i++;
            }
            //display the maximum
            printf("\nMaximum number = %d",max);
}
else if(option==7)
{
            total=0;

            //call function to read data from file
            total=ReadFile(filearray);

            //show sub menu
            do
            {
                  printf("\na). Print out the values");
                  printf("\nb). Find the maximum value");
```

Writingsol

```
printf("\nc). Calculate the RMS (Root Means Square)");
printf("\nd). Count the number of Negative values");
printf("\ne). Exit back to main menu");
fflush(stdin);
//ask for the sub choice
printf("\nPlease select option (a,b,c,d,e) =\t");
scanf("%c",&option2);

if(option2=='a')
{
	ShowArray(filearray,total);
}
else if(option2 == 'b')
{
	printf("\nThe maximum value is
%d",FindMaximum(filearray,total));
}
else if(option2 == 'c')
{
	printf("\nThe RMS (Root Mean Square ) is
%f",CalculateRMS(filearray,total));
}
else if(option2 =='d')
{
	printf("\nThe count of the negative numbers is
%d",CountNegative(filearray,total));
}
else if(option2 =='e')
{
}
```

Writingsol

```
                        else
                        {
                                printf("Invalid option please try again...");
                        }
                }
                while(option2!='e');
        }
        else if(option==8)
        {
                printf("\nGood Bye");
        }
        else
        {
                printf("\nInvalid option please try again...\n");
        }
}
while(option!=8);

//end the program
return 0;
}
```

Writingsol