Day1

1. Write a program to determine the roots of a quadratic equation ax2 + bx + c = 0. Your program should ask for the values of a, b and c, and print the roots (real or complex).

Ans:

```
#include<stdio.h>
#include<math.h>
int main()
{
	float a,b,c;
	printf("\nEnter the value of a b c:");
	scanf("%f",&a);
	scanf("%f",&b);
	scanf("%f",&c);
	float root1,root2;
	float D=b*b-4*a*c;
	if(D>0)
	{
		root1=(-b+sqrt(D))/(2*a);
		root2=(-b-sqrt(D))/(2*a);
		printf("The roots are %f and %f",root1,root2);
	}
	else if(D==0)
	{
		root1=(-b/(2*a));
		printf("The roots are %f and %f",root1);
	}
	else
	{
		float m=-D;
		m=sqrt(m);
		float c=-b/(2*a);
		printf("The roots are %f+i%f and %f-i%f",c,m/(2*a),c,m/(2*a));
	}
	return 0;
}
```

2. Read a sequence of positive integers a0, a1, a2, . . . (the length of the sequence will not be known a priori) and determine max i ∑ 4 j=0 ai+j . Note that you do not need to store the complete sequence in order to compute the required quantity.

Ans:

```
#include<stdio.h>
int main()
{
```

```
	int a,b,c,d,e;
	int max=-1;
	scanf("%d",&a);
	scanf("%d",&b);
	scanf("%d",&c);
	int sum;
	while(0!=scanf("%d",&d))
	{
	sum=a+b+c+d;
	max=(sum>max)?sum:max;
	a=b;
	b=c;
	c=d;
	}
	printf("\nThe maximum is %d",max);
	return 0;
}
```

3.Write a C program that can add one positive integer to another positive integer, and subtract one positive integer from another positive integer. The input integers may contain up to 100 digits (0–9). Thus, you will not be able to use the usual builtin types (int, long int, etc.) to store the inputs. You may not use the GNU Multiple Precision (MP) Arithmetic Library, or any other similar library or C++ class.

Ans:

```
#include<stdio.h>
#include<string.h>
void reverse(char* str)
{
	int len=strlen(str);
	int i;
	for(i=0;i<len/2;i++)
	{
		char temp;
		temp=str[i];
		str[i]=str[len-i-1];
		str[len-i-1]=temp;
	}
}
int main()
{
	char num1[100],num2[100],result[101];
	scanf("%s",num1);
	scanf("%s",num2);
	char op,a;
```

```
scanf("%c",&a);
scanf("%c",&op);
if(op=='+')
{
	reverse(num1);
	reverse(num2);
	int len1=strlen(num1),len2=strlen(num2);
	int i=0,j=0,carry=0,k=0;
	while(i<len1&&j<len2)
	{
		int digit1=(num1[i++]-'0');
		int digit2=(num2[j++]-'0');
		int sum=digit1+digit2+carry;
		int rem=(sum%10);
		carry=sum/10;
		result[k++]=('0'+rem);
	}
	if(i==len1)
	{
		while(j<len2)
		{
			int digit=(num2[j++]-'0');
			int sum=digit+carry;
			int rem=sum%10;
			carry=sum/10;
			result[k++]=rem+'0';
		}
	}
	if(j==len2)
	{
		while(i<len1)
		{
			int digit=(num1[i++]-'0');
			int sum=digit+carry;
			int rem=sum%10;
			carry=sum/10;
			result[k++]=rem+'0';
		}
	}
	if(carry!=0)
	{
		result[k]=(carry+'0');
	}
	int len=strlen(result);
```

```
        i=len-2;
        while(i>=0)
        {
                printf("%c",result[i]);
                i--;
        }

}
else if(op=='-')
{
        reverse(num1);
        reverse(num2);
        int len1=strlen(num1),len2=strlen(num2);
        int i=0,j=0,carry=0,k=0;
        while(i<len1&&j<len2)
        {
                int digit1=(num1[i++]-'0');
                int digit2=(num2[j++]-'0');
                int sum;
                if((digit1+carry)>=(digit2))
                {
                        sum=digit1+carry-digit2;
                        carry=0;
                }
                else if((digit1+carry)<digit2)
                {
                        sum=10+(digit1+carry)-digit2;
                        carry=-1;
                }
                int rem=(sum%10);
                result[k++]=('0'+rem);
        }
        if(i==len1)
        {
                while(j<len2)
                {
                        int digit=(num2[j++]-'0');
                        int sum;
                        if((carry-digit)>=0)
                        {
                                sum=carry-digit;
                                carry=0;
                        }
                        else
```

```
                        {
                                sum=10+carry-digit;
                                carry=-1;
                        }
                        int rem=sum%10;
                        result[k++]=rem+'0';
                }
        }
        if(j==len2)
        {
                while(i<len1)
                {
                        int digit=(num1[i++]-'0');
                        int sum;
                        if(carry+digit>=0)
                        {
                                sum=carry+digit;
                                carry=0;
                        }
                        else
                        {
                                sum=10+carry+digit;
                                carry=-1;
                        }
                        int rem=sum%10;
                        result[k++]=rem+'0';
                }
        }
        if(carry!=0)
        {
                result[k]=(-carry+'0');
                printf("-");
        }
        int len=strlen(result);
        i=len-1;
        while(i>=0)
        {
                printf("%c",result[i]);
                i--;
        }

    }

}
```

(some modification needed)

Day 2

1. Write a program that reads the given file (getc-input.txt) one character at a time using fgetc. After each character is read, print it along with its ASCII value to the screen. Try storing the return value of fgetc in an int type variable and a char type variable in turn, and report any observed difference in the behaviour of your program.

Ans:

```
#include<stdio.h>
#include<string.h>
int main()
{
	int c;
	while(EOF!=(c=fgetc(stdin)))
	{
		int x=(int)c;
		printf("%d ",x);
	}
	return 0;
}
```

3. Write a program that reads text typed at the terminal, and counts the number of occurrences of vowels in the input text; the number of words in the input text. Assume that any contiguous sequence of letters and / or digits forms one word.

Ans:

```
#include<stdio.h>
#include<string.h>
#include<string.h>
#include<ctype.h>
int main()
{
	int inword=0;
	int vowel_cnt=0;
	int word_cnt=0;
	int  c;
	while(EOF!=(c=fgetc(stdin)))
	{
		if(isalnum(c))
		{
			if(c=='A'||c=='a'||c=='E'||c=='e'||c=='I'||c=='i'||c=='O'||c=='o'||c=='U'||c=='u')
			{
				vowel_cnt++;
			}
			inword=1;
```

```
		}
		else
		{
			if(inword==1)
			{
				word_cnt++;
			}
			inword=0;
		}
	}
	printf("The vowel count is %d and the word count is %d",vowel_cnt,word_cnt);
	return 0;
}
```

Day3

(a) Write a program to convert a given string s to s ′ , its run-length encoded form. This means that s ′ will contain the same sequence of distinct characters as s, but any m > 1 consecutive occurrences of a character will be replaced by a single occurrence of the character immediately followed by the integer m (in base 10). For example, aaabccd should be converted to a3bc2d. The string s should be read from the terminal. It may contain letters (no digits), blanks and tabs (’\t’), but no newline. The length of the string s will not be known to you in advance

(b) Add code to your program so that it prints the character that occurs consecutively the maximum number of times. For the example above, your program should print a. If the maximum number of consecutive occurrences is the same for two or more characters, you may print any one

Ans:

```
#include<stdio.h>
#include<string.h>
int main()
{
	int c;
	char curr;
	int count=-1;
	int max_count=-1;
	char max_char;
	while(EOF!=(c=fgetc(stdin)))
	{
		if(c==' '||c=='\t')
		{
			continue;
		}
		if(curr!=c)
		{
			if(count==-1)
			{
				curr=c;
				count=1;
			}
			else
			{
			if(count==1)
			{
				printf("%c",curr);
```

```
                        }
                        else
                        {
                                printf("%c%d",curr,count);
                        }
                        if(count>max_count)
                        {
                                max_count=count;
                                max_char=curr;
                        }
                        curr=c;
                        count=1;

                        }
                }
                else
                {
                        count++;
                }
        }
        printf("\nThe maximum count of the charecter is %d and the charecter is
%c",max_count,max_char);
        return 0;
}
```

(c) Modify your program so that it decodes a given run-length encoded string s ′ to its original form s. For example, given a3bc2d, your program should print aaabccd. Note that, given abcd, your program should print abcd itself. The input format will be the same as for part (a). NOTE: You do not need to use an array for this problem.

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int main()
{
        int c;
        char current,next,prev,update=0;
        while(EOF!=(c=fgetc(stdin)))
        {
                if(isalnum(c)==0)
                {
                        continue;
                }
                if(('a'<=c)&&(c<='z'))
                {
                        current=c;
```

```
                update=1;
            }
            else if(('0'<=c)&&(c<='9'))
            {
                int num=c-'0';
                int i=0;
                for(i=0;i<num-1;i++)
                {
                    printf("%c",current);
                }
                update=0;
            }
            if(update==1)
            {
                printf("%c",current);
            }
        }
        return 0;
}
```

2 Write a program that reads text typed at the terminal, and counts the number of occurrences of each of the letters a–z in the text. For this problem, you should not distinguish between uppercase and lowercase letters. The input text may span multiple lines, and may contain digits, punctuation marks, blanks, tabs, and other printable characters, which you should ignore.

Ans:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int isdigit(char c)
{
        if(('0'<=c)&&(c<='9'))
        {
            return 1;
        }
        else
        {
            return 0;
        }
}
int main()
{
        int c;
        char current,next,prev,update=0;
        int arr[26]={0};
```

```
	while(EOF!=(c=fgetc(stdin)))
	{
		if(isdigit(c)==1)
		{
			continue;
		}
		else if(isalnum(c)==0)
		{
			continue;
		}
		else
		{
			if(('A'<=c)&&(c<='Z'))
			{
				arr[c-'A']++;
			}
			else if(('a'<=c)&&(c<='z'))
			{
				arr[c-'a']++;
			}
		}

	}
	int i;
	for(i=0;i<26;i++)
	{
		printf("\ncharecter %c count: %d",'a'+i,arr[i]);
	}
	return 0;
}
```

3 Write a program that takes a single string as input, reverses the string in place and prints the reversed string to the terminal. You are told that the input string will contain at most 80 characters, and will not contain any whitespace (blanks, tabs or newlines).

Ans:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
	char str[80];
	int i,c;
	while(EOF!=(c=fgetc(stdin)))
	{
		str[i++]=c;
```

```
	}
	int len=i;
	for(i=0;i<len/2;i++)
	{
		char temp;
		temp=str[i];
		str[i]=str[len-i-1];
		str[len-i-1]=temp;
	}
	for(i=0;i<len;i++)
	{
		printf("%c",str[i]);
	}
	return 0;
}
```

4 Given an array of at most 100 integers, print the longest sequence of (a) elements that appear in ascending order; (b) consecutive elements that appear in ascending order.
Ans:
a)

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
	int arr[100];
	int c,i=0;
	while(0!=scanf("%d",&c))
	{
		arr[i++]=c;
	}
	int len=i;
	int max_count=-1;
	int count=0;
	i=0;
	int j;
	for(i=0;i<=len-2;i++)
	{
		int pivot=arr[i];
		for(j=i+1;j<len;j++)
		{

			if(arr[j]>pivot)
			{
				pivot=arr[j];
```

```
                        count++;
                    }
                }
                max_count=(count>max_count)?count:max_count;
                count=0;
        }
        printf("\nThe maximum contigious increasing subarray length is %d",max_count+1);
        return 0;
}
```

b)

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
        int arr[100];
        int c,i=0;
        while(0!=scanf("%d",&c))
        {
                arr[i++]=c;
        }
        int len=i;
        int max_count=-1;
        int count=0;
        i=0;
        while(i<=len-1)
        {
                if(arr[i]<arr[i+1])
                {
                        count++;
                }
                else
                {
                        max_count=(count>max_count)?count:max_count;
                        count=0;
                }
                i++;
        }
        printf("\nThe maximum contigious increasing subarray length is %d",max_count+1);
        return 0;
}
```

Day4

1. Consider 2 sequences of letters (a–z), A and B, stored in arrays. (a) Write a program to find the number of (possibly overlapping) occurrences of the sequence B in A. (b) Write a program to find whether the multisets corresponding to A and B are equal.

Ans:

a)

```
#include<stdio.h>
#include<string.h>
int compare(char* a,char* b,int k,int m)
{
	int i;
	for(i=0;i<m;i++)
	{
		if(a[i+k]!=b[i])
		{
			return 0;
		}
	}
	return 1;
}
int main()
{
	char a[100];
	char b[50];
	scanf("%s",a);
	scanf("%s",b);
	int len1=strlen(a);
	int len2=strlen(b);
	int i,j;
	int count=0;
	for(i=0;i<=len1-len2;i++)
	{
		if(compare(a,b,i,len2)==1)
		{
			count++;
		}
	}
	printf("The number of occurences of the B string in A string is %d",count);
	return 0;
}
```

b)

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

```
#include<stdlib.h>

#define CHAR_256 123
int main()
{
   int freqA[CHAR_256];
   int freqB[CHAR_256];

   //printf("%d",'z');
   memset(freqA,0,123*sizeof(int));
   memset(freqB,0,123*sizeof(int));

   char * A = (char *)malloc(100*sizeof(char));
   char * B = (char *)malloc(100*sizeof(char));
   printf("Enter the first array : ");
   scanf("%s",A);
   printf("\nEnter the second array : ");
   scanf("%s",B);
   for(int i = 0;i<strlen(A);i++)
   {
      freqA[A[i]]++;
   }
   for(int i = 0;i<strlen(B);i++)
   {
      freqB[B[i]]++;
   }
   if(strlen(A) != strlen(B))
   {
      printf("\nMultisets of A and B are not equal.");
   }
   else
   {
      int flag = 1;
      for(int i = 0;i<strlen(A);i++)
      {
         if(freqA[A[i]] != freqB[A[i]])
         {
            flag = 0;
            break;
         }
      }
      if(flag == 1)
      {
         printf("\nMultisets of A and B are equal.");
```

```
        }
        else
        {
            printf("\nMultisets of A and B are not equal.");
        }
    }
}
```

2. Write a program that first reads multiple lines of text from the terminal, and then, depending on the user's choice, prints either the odd- or the even-numbered lines, either in their original or in reverse order. You may assume that lines are numbered starting with one; each line is no more than 80 characters long; the input text will not consist of more than 10 lines. Redesign your program so that it will run correctly even if the number of lines in the input text is not known a priori.

Ans:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    // Dynamically allocate an array of pointers for lines
    char **arr;
    int max_lines = 10;
    int max_chars = 80;
    arr = (char **)malloc(max_lines * sizeof(char *));

    // Allocate memory for each line
    int i;
    for ( i = 0; i < max_lines; i++)
    {
        arr[i] = (char *)malloc(max_chars * sizeof(char));
    }

    // Read lines and store their length
    int length[max_lines];
    int count = 0;
    int line_count = 0;

    printf("Enter up to 10 lines of text (each no more than 80 characters):\n");

    while (line_count < max_lines && fgets(arr[line_count], max_chars, stdin))
```

```
{
    // Remove the newline character from the end of the line if present
    int len = strlen(arr[line_count]);
    if (arr[line_count][len - 1] == '\n')
    {
        arr[line_count][len - 1] = '\0';
        len--;
    }
    length[line_count] = len;
    line_count++;
}

// Get user input for choices
printf("\nEnter your choice: 1. Print even-numbered lines  2. Print odd-numbered lines: ");
int choice1;
scanf("%d", &choice1);

printf("Print the lines in: 1. Reverse order  2. Original order: ");
int choice2;
scanf("%d", &choice2);

// Print even or odd lines based on user choice
int start = (choice1 == 1) ? 1 : 0; // Start at 1 for even, 0 for odd
for ( i = start; i < line_count; i += 2)
{
    if (choice2 == 1)
    {
        int j;
        // Print in reverse order
        for ( j = length[i] - 1; j >= 0; j--)
        {
            printf("%c", arr[i][j]);
        }
    }
    else
    {
        // Print in original order
        printf("%s", arr[i]);
    }
    printf("\n");
}

// Free allocated memory
for (i = 0; i < max_lines; i++)
```

```
    {
        free(arr[i]);
    }
    free(arr);

    return 0;
}
```

Day5

1. Implement a list of integers using dynamic memory. The list consists of a header storing two items: the size of the list, and a dynamically allocated array capable of storing only the elements in the list. Define a structure to store the size of the list, and a dynamically allocated array that stores the list elements. Use a typedef to define LIST as the name of your structure. Implement the following functions. (a) create_list(void): returns an empty list. For an empty list, the size is zero, and the array is NULL. (b) print_list(LIST L): print the elements of the list separated by spaces, and terminated by a newline. Computing Lab (ISI) Functions 17 / 23 Practice problems II (c) append(LIST L, int a): appends a to the end of the list L, and returns the modified list. (d) prepend(LIST L, int a): prepends a at the beginning of the list L, and returns the modified list. (e) deletelast(LIST L): deletes the last element of the list, and returns the modified list. (f) deletefirst(LIST L): deletes the first element of the list, and returns the modified list. (g) deleteall(LIST L, int a): deletes all occurrences of a in L, and returns the modified list. Do not forget to use free() where necessary

Ans:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct linkedlist
{
        int n;
        char** a;
}list;
list head;
int first=-1;
int last=-1;
void createlist()
{
        head.n=10;
        char** b;
        int i;
        for(i=0;i<head.n;i++)
        {
                b[i]=(char*)malloc(9*sizeof(char));
        }
        for(i=0;i<head.n;i++)
        {
                b[i]=NULL;
        }
        head.a=b;
}
void printlist()
```

```
{
	int num=head.n;
	printf("(");
	int i=0;
	for(i=0;i<num;i++)
	{
		if(head.a[i]!=NULL)
		{
			printf("%s,",head.a[i]);
		}
	}
	printf(")");
	printf("\n");
}
void append(char* a)
{
	if(first==-1&&last==-1)
	{
		first++;
		last++;
		head.a[last]=a;
	}
	else
	{
		last++;
		head.a[last]=a;
	}
	printlist();
}
void prepend(char* a)
{
	int i;
	for(i=last+1;i>first;i--)
	{
		head.a[i]=head.a[i-1];
	}
	head.a[0]=a;
	first=0;
	last++;
	printlist();
}
void deletelast()
{
	head.a[last]=NULL;
```

```
	last--;
	printlist();
}
void deletefirst()
{
	int i;
	for(i=first;i<last;i++)
	{
		head.a[i]=head.a[i+1];
	}
	head.a[last]=NULL;
	last--;
	printlist();
}
void delete(int index)
{
	int i;
	for(i=index;i<last;i++)
	{
		head.a[i]=head.a[i+1];
	}
	head.a[last]=NULL;
	last--;
}
void deleteall(char* b)
{
	int i;
	for(i=first;i<=last;i++)
	{
		if(strcmp(head.a[i],b)==0)
		{
			delete(i);
		}
		if(head.a[i]==NULL)
		{
			break;
		}
	}
	printlist();
}
int main()
{
	createlist();
	printlist();
```

```
        append("apple");
        append("lemon");
        append("banana");
        append("pineapple");
        append("pear");
        append("apple");
        append("orange");
        append("cherry");
        append("mango");
        prepend("Durga");
        deletelast();
        deletefirst();
        deleteall("apple");
        return 0;
}
```

2. Let s and t be strings. Implement the following functions in C: (a) strlen(s): returns the length of s, i.e., the number of characters present in s; (b) strcmp(s, t): returns 1 if s and t are identical, 0 otherwise; (c) diffByOne(s, t): returns 1 if s and t are of the same length, and differ in exactly one position, 0 otherwise. Thus, if s is sale and t is either salt or same or pale, diffByOne(s, t) should return 1; but if s and t are salt and salty, it should return 0.

Ans:

```
#include<stdio.h>
int stlen(char* s)
{
        int length=0;
        int i=0;
        while(s[i]!='\0')
        {
                i++;
                length++;
        }
        return length;
}
int strcmp(char* s,char* t)
{
        int count=0;
        while(*s==*t)
        {
                count=1;
                s++;
                t++;
        }
        return count;
```

```
}
int diffByOne(char* s,char* t)
{
	int diff;
	if(stlen(s)!=stlen(t))
	{
		return 0;
	}
	else
	{
		int i=0;
		int len=stlen(s);
		 diff=0;
		for(i=0;i<len;i++)
		{
			if(s[i]!=t[i])
			{
				diff++;
			}
		}
	}
	if(diff==1)
	{
		return 1;
	}
	else
	{
		return 0;
	}
}
int main()
{
	char a[100];
	char b[100];
	scanf("%s",a);
	scanf("%s",b);
	printf("Result of strlen is %d and %d",stlen(a),stlen(b));
	printf("\nResult of strcmp is %d",strcmp(a,b));
	printf("\nThe result of diffbyone is %d",diffByOne(a,b));
	return 0;
}
```

3. Write a function uniquify that takes two arguments: an array A containing non-negative integers, and n, the number of integers contained in the array, and stores in A a list of the

distinct integers contained in A if the integers in A are sorted in increasing order, and returns the number of distinct integers in A; and returns -1 otherwise.

Ans:

```
#include<stdio.h>
#include<stdlib.h>
int uniquify(int* arr,int n)
{
	int i=0;
	for(i=0;i<=n-2;i++)
	{
		if(arr[i]>arr[i+1])
		{
			return -1;
		}
	}
	int count=0;
	int b[n];
	for(i=0;i<n-1;i++)
	{
		if(arr[i]!=arr[i+1])
		{
			b[count++]=arr[i];

		}
	}
	if(arr[n-1]!=arr[n-2])
	{
		b[count++]=arr[n-1];
	}
	for(i=0;i<count;i++)
	{
		arr[i]=b[i];
	}
	return count;
}
int main()
{
	int len;
	printf("Enter the length of the array:");
	scanf("%d",&len);
	int i;
	int* arr=(int*)malloc(len*sizeof(int));
	printf("\nEnter the array elements:");
	for(i=0;i<len;i++)
```

```
        {
                int x;
                scanf("%d",&x);
                arr[i]=x;
        }
        int p=uniquify(arr,len);
        printf("The number of unique elements is %d",p);
        if(p!=-1)
        {
        printf("\nPrinting the unique elements:");
        for(i=0;i<p;i++)
        {
                printf("%d ",arr[i]);
        }
        }
        return 0;
}
```

4. Recall that the rand() function provided by the standard C library generates a sequence of pseudo-random integers uniformly sampled from the range [0, RAND_MAX]. For the first part of this question, you have to use the Box-Muller transform (described below) to generate a pseudo-random sequence drawn from a normal distribution. [5] Suppose u1 and u2 are independent samples chosen from the uniform distribution on the unit interval (0, 1). Let z0 = √ −2 ln u1 cos(2πu2) z1 = √ −2 ln u1 sin(2πu2) Then z0 and z1 are independent random variables with a standard normal distribution.
Write a function that uses the above idea to generate a pseudo-random sequence drawn from a standard normal distribution. Your function should have the following prototype: double normal(void);
Ans:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

typedef struct {
   double num1;
   double num2;
} Rnd_Num;

Rnd_Num normal(void)
{
   double u1 = (rand() + 1.0)/ (RAND_MAX + 2.0);
   double u2 = (rand() + 1.0) / (RAND_MAX + 2.0);

   double z0 = sqrt(-2.0 * log(u1)) * cos(2.0 * 3.14 * u2);
```

```
    double z1 = sqrt(-2.0 * log(u1)) * sin(2.0 * 3.14 * u2);

    Rnd_Num st;
    st.num1 = z0;
    st.num2 = z1;
    return st;
}

int main() {
    srand(time(0));

    //printf("RAND_MAX is: %d\n", RAND_MAX);

    printf("Normal Number Sequence are \n");
    for (int i = 0; i < 100; i++)
    {
        Rnd_Num rd = normal();
        printf("Num1 and Num2: %f and %f \n",rd.num1,rd.num2);
    }

    return 0;
}
```

5. A person's daily commute from home to office consists of three stages: she first takes an auto to the nearest Metro station A, then takes the Metro from station A to station B, and finally takes a bus from B to her office. Assume that the time taken to complete each stage follows an independent normal distribution N(μi , σ2 i) for i = 1, 2, 3. You have to write a program that uses simulation to estimate the probability that the person's commute will take more than a specified amount of time T. Input format. The values of μ1, σ1, μ2, σ2, μ3, σ3 and T (all in minutes) will be provided in that order via standard input. Output format. Your program should print a single floating point number corresponding to the desired probability, correct to 8 decimal places.

Ans:

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
typedef struct
{
        double num1;
        double num2;
}Rnd_Num;
Rnd_Num normal(void)
{
    double u1 = (rand() + 1.0)/ (RAND_MAX + 2.0);
```

```
    double u2 = (rand() + 1.0) / (RAND_MAX + 2.0);

    double z0 = sqrt(-2.0 * log(u1)) * cos(2.0 * 3.14 * u2);
    double z1 = sqrt(-2.0 * log(u1)) * sin(2.0 * 3.14 * u2);

    Rnd_Num st;
    st.num1 = z0;
    st.num2 = z1;
    return st;
}
double normalCDF(double z) {
        int i=1;
        int count=0;
        for(i=1;i<=5000;i++)
        {
                Rnd_Num st=normal();
                if(st.num1<=z)
                {
                        count++;
                }
                if(st.num2<=z)
                {
                        count++;
                }
        }
        double value=(double)count/10000;
        return value;
}
int main()
{
        double mu1,sigma1,mu2,sigma2,mu3,sigma3;
        printf("\nEnter the average and standard deviation of first way:");
        scanf("%lf",&mu1);
        scanf("%lf",&sigma1);
        printf("\nEnter the average and standard deviation of second way:");
        scanf("%lf",&mu2);
        scanf("%lf",&sigma2);
        printf("\nEnter the average and standard deviation of third way:");
        scanf("%lf",&mu3);
        scanf("%lf",&sigma3);
        double T;
        printf("\nEnter the value of T:");
        scanf("%lf",&T);
        double mean=mu1+mu2+mu3;
```

```
	double variance=(sigma1*sigma1)+(sigma2*sigma2)*(sigma3*sigma3);
	double sd=sqrt(variance);
	double z_value=(T-mean)/sd;
	if(z_value>=0)
	{
		printf("\nThe probability to reach is %.8lf",1-normalCDF(z_value));
	}
	else
	{
		printf("\nThe probability to reach is %.8lf",normalCDF(-z_value));
	}
	return 0;
}
```