Assignment:

OOPD:

By Surbhi , MT19055

1. Roots of Quadratic Equation
   (a) Without structuring program:

```
#include <stdio.h>
#include<math.h>
int main()
{
   int a, b,c,real,imaginary;
   int d,d1;
   float r1,r2;
   printf("Enter a,b,c: ");
   scanf("%d%d%d",&a,&b,&c);
   d=b*b-(4*a*c);
   d1=sqrt(d);
   if(d==0)
     {
      printf("roots are equal and real");
      r1 = r2 = -b/(2*a);
      printf("r1 = r2= %.2f", r1);

     }
   else if(d>0){
         printf("roots are real and distinct");
         r1=(-b+d1)/(2*a);
         r2=(-b-d1)/(2*a);
         printf("r1 = %.2f and r2= %.2f", r1, r2);
   }
    else
        {
         printf("real and imaginary roots");
```

```
        }
    return 0;
}
```

(b) Using Structure Programming:

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

void calculateRoots(int a, int b, int c)
{
   int d, d1;
   float r1,r2;
   d=b*b-(4*a*c);
   d1=sqrt(d);
   if(d==0)
     {
      printf("roots are equal and real");
      r1 = r2 = -b/(2*a);
      printf("r1 = r2= %.2f", r1);
     }
   else if(d>0){
         printf("roots are real and distinct");
         r1=(-b+d1)/(2*a);
         r2=(-b-d1)/(2*a);
         printf("r1 = %.2f and r2= %.2f", r1, r2);
   }
    else
        {
         printf("imaginary roots");
        }
}
void main(){
           int a, b , c;
            printf("Enter a,b,c: ");
            scanf("%d%d%d" ,&a,&b,&c);
            calculateRoots(a,b,c);     }
```

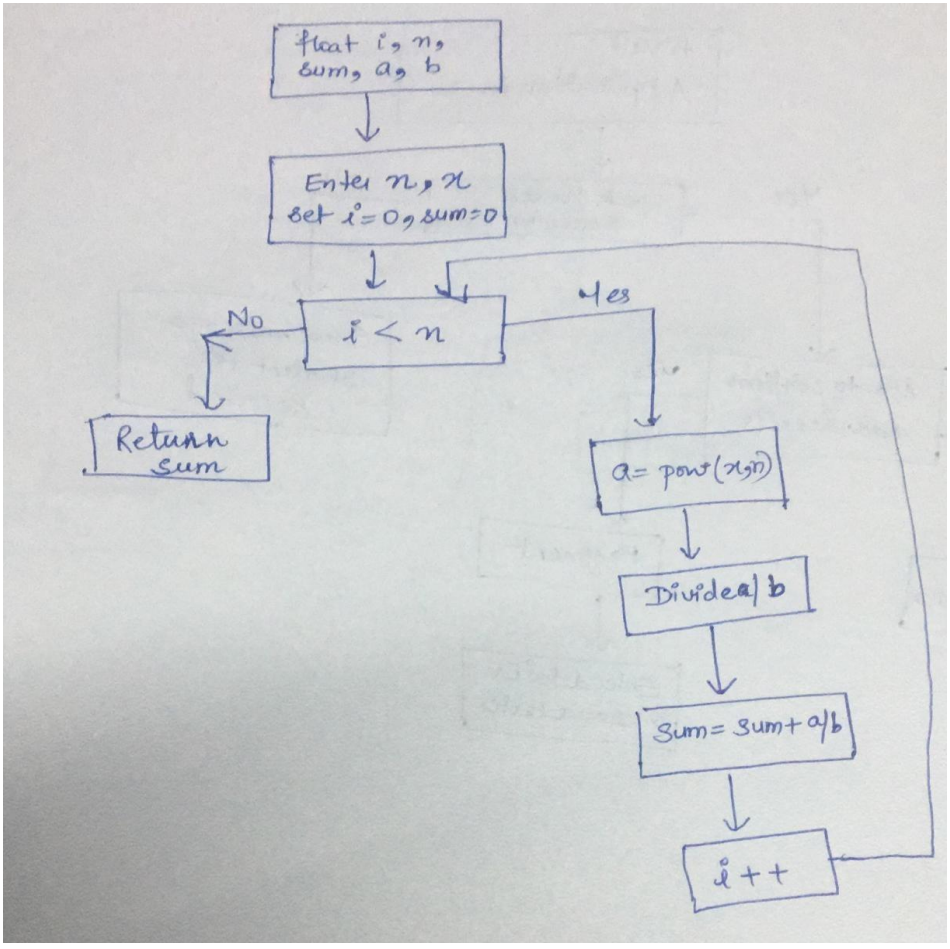
Advantages:

- It is user friendly and easy to understand.
- Maintenance is easy.
- It's easy to find bugs.
- It has reuse ability.

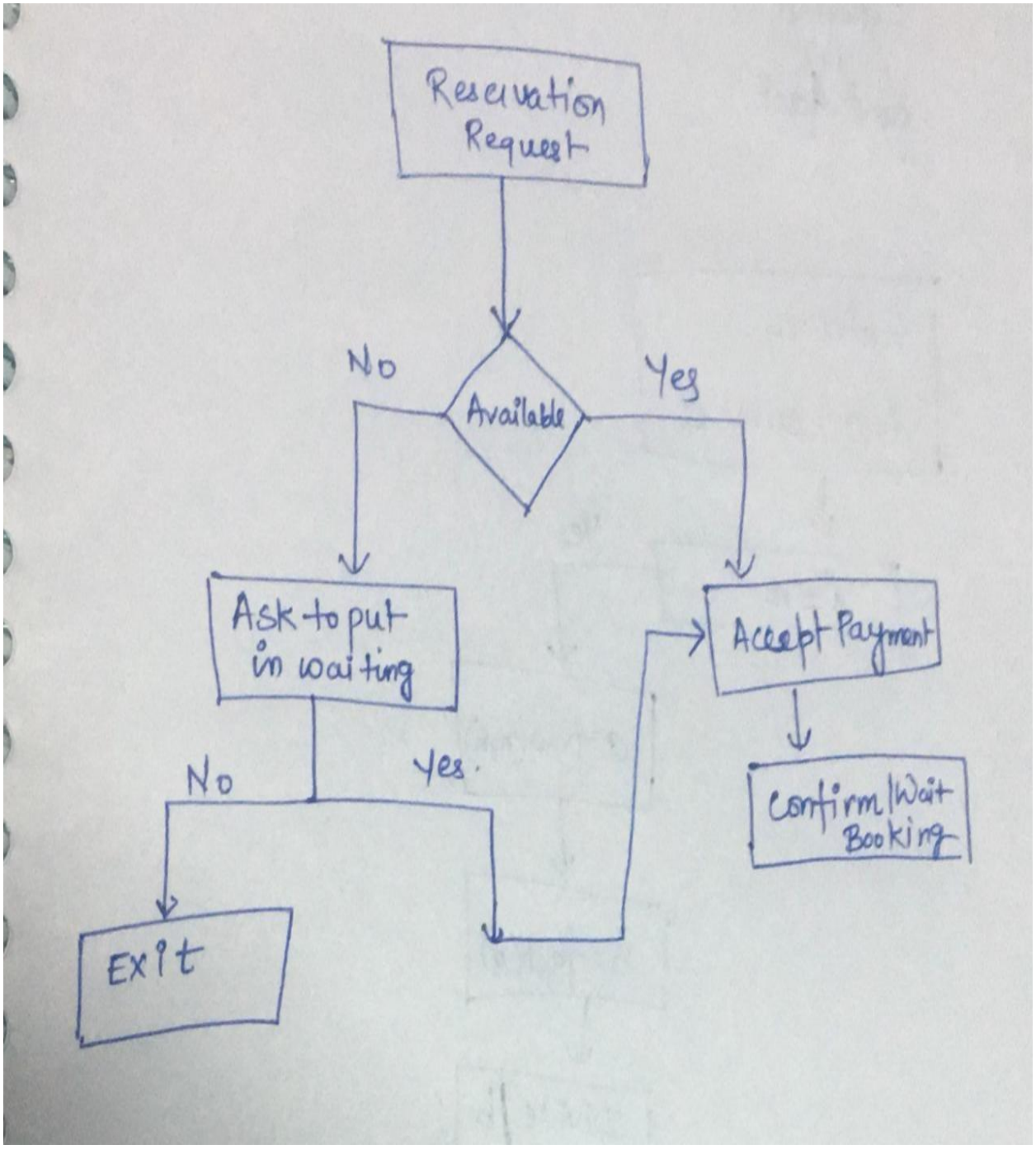
Disadvantages:

- It has no encapsulation.
- Lack of data hiding.

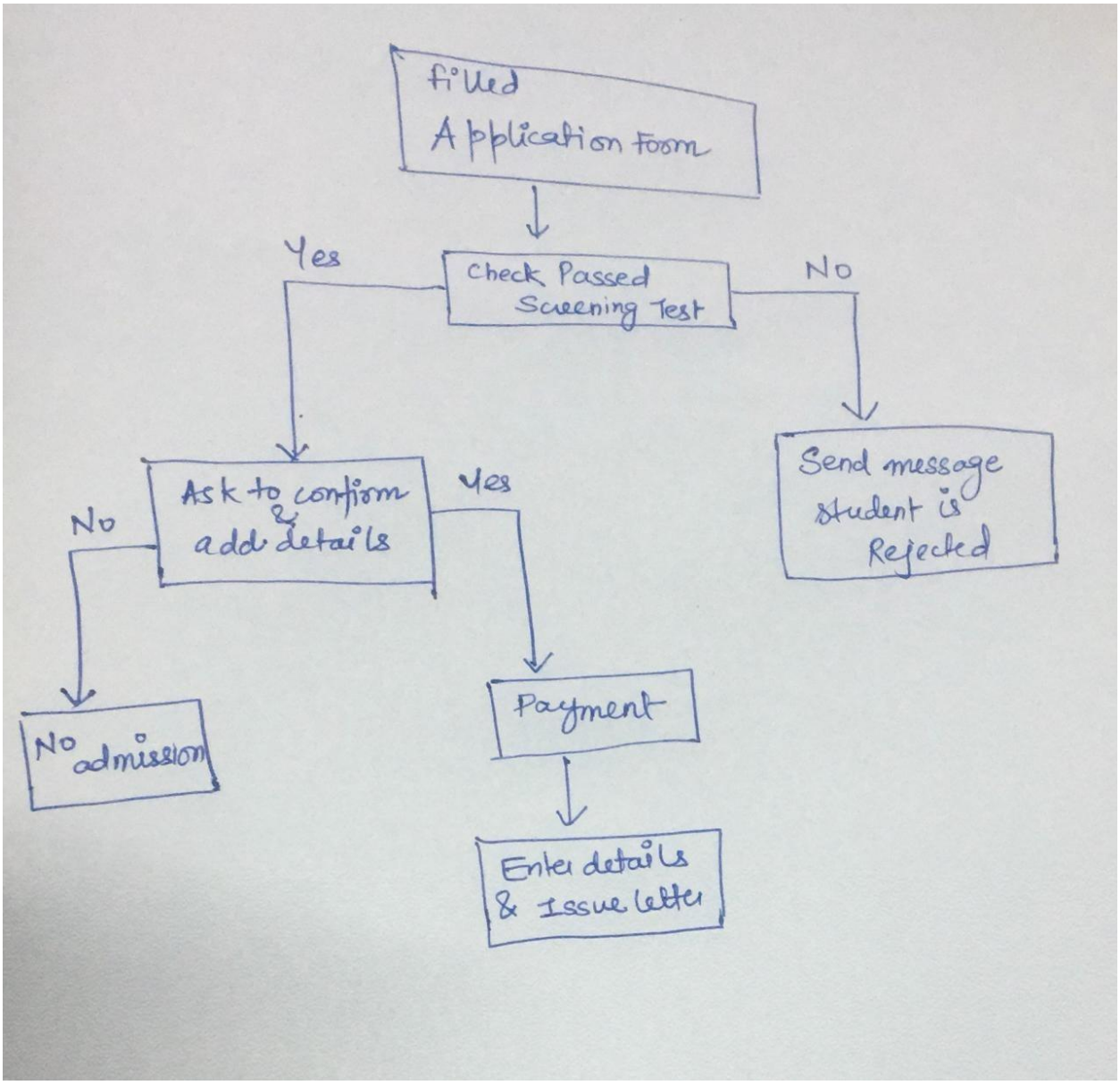
2. Program structure
   (a) Program structure of mathematical expression



(b) Railway Reservation



(c) Admission Process



3. a) Program to find total salary of employees; given total number of employees are 5 and structure programming is not allowed.

```
#include <stdio.h>

#include <stdlib.h>

int main()

{

   int i,n,total_sal=0,salary;

   printf("Enter the number of employees\n");

   scanf("%d",&n);

   for(i=0;i<n;++i)

  { printf("Enter salary of id %d :",i+1);

    scanf("%d",&salary);

    total_sal= total_sal +salary;

  }

 printf("Total Salary of the %d Employees = %d",n, total_sal);

 return 0;

}
```

3. b) Using arrays

```
#include<stdio.h>

#include<stdlib.h>

int main()

{

int sal[10], i, n, total_sal = 0;
```

```
printf("Enter nuber of employees: ");
scanf("%d", &n);
for(i=0; i<n; ++i)
{
printf("Enter salary of id%d: ",i+1);
scanf("%d", &sal[i]);
total_sal+= sal[i];
}
printf("Total Salary of the %d Employees = %d",n, total_sal);
return 0;
}
```

3. c) Arrays are more user friendly. When the number of employees increases then it become a tedious task to maintain details of all the employees using 3(a) approach as it'll be difficult to understand and it'll be more prone to errors.

3. d) Using Structures:

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

struct emp{
   char name[10];
   int id;
   float sal;
   };
```

```
int main()
{
        struct emp e[5];
        int n,i, total_sal=0;
        printf("Enter the number of employees: ");
        scanf("%d",&n);
        for(i=0; i<n; i++)
        {
                printf("Name: ");
                scanf("%s",&e[i].name);
                printf("Id: ");
                scanf("%d",&e[i].id);
                printf("Salary: ");
                scanf("%f",&e[i].sal);
        }
        for(i=0;i<n;i++)
        {
                total_sal=total_sal+e[i].sal;
        }
        printf("Total Salary of the %d Employees = %d",n, total_sal);
        return 0;
        }
```

3. e) Structure is a collection of heterogeneous data type i.e. structures can have different type of variables stored in it like int,

float, char, string, array etc. whereas arrays can store values of same data type only.

.

4. Hash table can be used here as here, key represent the id of the employee and rest of the information about employee can be stored in the 'value' part. Also insertion and deletion in hash tables is easy.

5. Organisation has 10 employees whose salary is available in array.

   (a) Print the sum of salaries of those employees who earn more than Rs. 50,000.

```
#include<stdio.h>

#include<stdlib.h>


void main(){

int arr[10]={10000,80000,35000,60000,40000,52900,70000,82100,29000,90400};

int i=0;

int sum=0;

repeat:

        if(i<=10)

        {

                i++;

                if(arr[i-1]>50000)

                {

                        sum = sum + arr[i-1];
```

```
                goto repeat;
        }
        else
        {
                goto repeat;
        }
    }
  printf("sum is %d",sum);
}
```

(b) Find the first employee with salary > Rs. 50,000 and print the salary value.

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

int main()
{
int arr[10]={10000,80000,35000,60000,40000,52900,70000,82100,29000,90400};
int i=0;
int flag;

repeat:
    if(i<=10)
    {
```

```
                i++;
                if(arr[i-1]>50000)
                {
                        flag = i-1;
                        goto end;
                }

                else
                {
                        flag = 21;
                        goto repeat;
                }
        }

end:
        if(flag!=21)
        {
        printf("Employee no.is %d ",flag+1);
        printf("Employee salary: %d ",arr[flag]);
        }

        else
        {
        printf("No employee");
```

```
        }
}
```

(c) Using For and while instead of goto:

(i) For

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

int main()
{
int arr[10]={10000,80000,35000,60000,40000,52900,70000,82100,29000,90400};
int i,sum=0;
   for(i=0;i<10;i++)
{
      if(arr[i]>50000)
         {
                  sum=sum+arr[i];
         }
}
   printf("sum is %d",sum);
   return 0;
}
```

(ii) While

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

```
int main()
{
   int arr[10]={10000,40000,35000,60000,40000,52900,70000,82100,29000,90400};
int i=0,temp;
while(i<10)
{
        if(arr[i]>50000)
        {
                temp=i;
                break;
        }
        i++;
}
   printf("Employee id is %d ",temp+1);
   printf("Employee salary is : %d ",arr[temp]);
   return 0;
}
```

(d) Disadvantages of goto statement:

- Make a mess of sequence of code
- Unintended infinite loop
- Reduces readability