Sahil Saini Salaria
Reg No. 180905048
Roll No. 11 C

Question1

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

int main(int argc, char* argv[]){
   int rank,size,fact=1,factsum,errclass,resultlen;
   char err_buffer[1000];
   MPI_Init(&argc,&argv);
   MPI_Errhandler_set(MPI_COMM_WORLD,MPI_ERRORS_RETURN);
   MPI_Comm_rank(MPI_COMM_WORLD,&rank);
   MPI_Comm_size(MPI_COMM_WORLD,&size);
   for(int i=1;i<=rank+1;i++){
      fact*=i;
   }
   int ierr=MPI_Scan(&fact,&factsum,1,MPI_INT,MPI_SUM,MPI_COMM_WORLD);
   if(ierr!=MPI_SUCCESS){
      MPI_Error_class(ierr,&errclass);
      if(errclass==MPI_ERR_COUNT){
         printf("Invalid rank used in MPI Scan call\n");
         MPI_Error_string(ierr,err_buffer,&resultlen);
         printf("%s",err_buffer);
         MPI_Finalize();
      }
   }
   if(rank==size-1){
      fprintf(stdout,"process %d factorial sum %d\n",rank,factsum);
      fflush(stdout);
   }
   MPI_Finalize();
   return 0;
}

// mpicc q1.c -lm -o q1 && mpirun -np 4 ./q1
```

Question 2

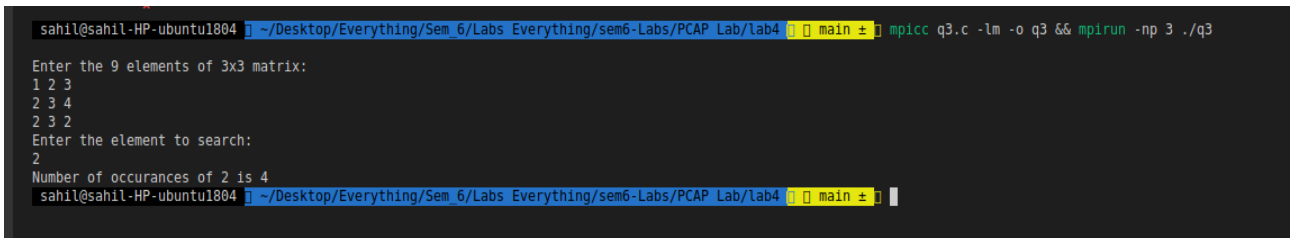
```
#include<stdio.h>
#include<mpi.h>
```

```
float func(float x){
   return 4/(1+x*x);
}
int main(int argc, char* argv[]){
   int rank,size;
   MPI_Init(&argc,&argv);
   MPI_Errhandler_set(MPI_COMM_WORLD,MPI_ERRORS_RETURN);
   MPI_Comm_rank(MPI_COMM_WORLD,&rank);
   MPI_Comm_size(MPI_COMM_WORLD,&size);

   float interval,mid,height,sum,area;
   int errclass;

   interval=(1/(float)size);
   mid=(rank*interval)+interval/2;
   height=func(mid);
   area=height*interval;
   int ierr=MPI_Reduce(&area,&sum,1,MPI_FLOAT,MPI_SUM,0,MPI_COMM_WORLD);

   if(ierr!=MPI_SUCCESS){
      MPI_Error_class(ierr,&errclass);
      if(errclass==MPI_ERR_COUNT){
         printf("Invalid rank used in MPI Scan call\n");

         MPI_Finalize();
      }
   }
   if(rank==0){
      printf("Process %d pi value is %f\n",rank,sum);
   }


   MPI_Finalize();
   return 0;
}

// mpicc q2.c -lm -o q2 && mpirun -np 4 ./q2
```

Question 3

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

```
int main(int argc, char* argv[]){
    int arr[3][3],size,rank,occurance=0,element,finsum,b[3],errclass;
    MPI_Init(&argc,&argv);
    MPI_Errhandler_set(MPI_COMM_WORLD,MPI_ERRORS_RETURN);
     MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);

    if(rank==0){
        printf("Enter the 9 elements: \n");
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                scanf("%d",&arr[i][j]);
            }
        }

        printf("Enter the element to search: \n");
        scanf("%d",&element);
    }
    MPI_Bcast(&element,1,MPI_INT,0,MPI_COMM_WORLD);
    int ierr=MPI_Scatter(arr,3,MPI_INT,b,3,MPI_INT,0,MPI_COMM_WORLD);

    if(ierr!=MPI_SUCCESS){
        MPI_Error_class(ierr,&errclass);
        if(errclass==MPI_ERR_COUNT){
            printf("Invalid rank used in MPI Scan call\n");

            MPI_Finalize();
        }
    }
    for(int i=0;i<3;i++){
        if(b[i]==element){
            occurance++;
        }
    }
    MPI_Scan(&occurance,&finsum,1,MPI_INT,MPI_SUM,MPI_COMM_WORLD);
    if(rank==size-1){
        printf("Number of occurances of %d is %d\n",element,finsum);
    }
    MPI_Finalize();
    return 0;


}

// mpicc q3.c -lm -o q3 && mpirun -np 4 ./q3
```



Question 4

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
void ErrorHandler(int error_code)
{
   if (error_code != MPI_SUCCESS)
   {
      char error_string[BUFSIZ];
      int length_of_error_string, error_class;
      MPI_Error_class(error_code, &error_class);
      MPI_Error_string(error_code, error_string, &length_of_uerror_string);
      printf("Errors:= %d \t %s\n", error_class, error_string);
   }
}
int main(int argc, char **argv)
{
   int rank, size, error_code;
   int arr[4][4], arr2[4], partsum[20], i, j, ele, pos;
   MPI_Init(&argc, &argv);
   MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);
   error_code = MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   ErrorHandler(error_code);
   error_code = MPI_Comm_size(MPI_COMM_WORLD, &size);
   ErrorHandler(error_code);
   if (rank == 0)
   {
      printf("Enter  4X4 matrix\n");
      for (i = 0; i < 4; i++)
      {
         for (j = 0; j < 4; j++)
         {
            scanf("%d", &arr[i][j]);
         }
      }
      printf("Ouput Matrix is: \n");
   }
   MPI_Scatter(arr, 4, MPI_INT, arr2, 4, MPI_INT, 0, MPI_COMM_WORLD);
   MPI_Scan(arr2, partsum, 4, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
   for (i = 0; i < 4; i++)
   {
      printf("%d ", partsum[i]);
```

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

    MPI_Finalize();
    return 0;
}
// mpicc q4.c -lm -o q4 && mpirun -np 4 ./q4
```