Programming in C/C++

LAB MANUAL

Lab Exercises for Week 1

Q1. Write a program to demonstrate the use of output statements that draws any object of your choice a. Draw a Christmas tree using '*' b. Draw a pyramid using '*'
Q2. Write a program that reads in a month number and outputs the month name.
Q3. Write a program to demonstrate the use of various input statements like getchar(), getch(),scanf()
Q4. Write a program to demonstrate the overflow and underflow of various data type and their resolution a. Demonstrate the overflow and underflow in integer data type b. Demonstrate the overflow and underflow in char data type c. Demonstrate the resolution of overflow and underflow
Q5. Write a program to demonstrate the precedence of various operators a. Demonstrate the precedence of arithmetic and unary operators b. Demonstrate the precedence of relational and logical operators

Q1. Write a program to demonstrate the use of output statements that draws any object of your choice

/* christmas tree*/

```
 #include<stdio.h>
  int main()
 {
      int i,l,k,m,j,rows,starNo,spaceNo;
      printf("Enter Rows:\n");
      scanf("%d",&rows);

      for(i=1;i<=rows;i++)
     {
        starNo=i*2-1;
            spaceNo=i+rows-starNo;
            for(j=0;j<spaceNo;j++)
        {
            printf(" ");
        }
            for(k=0;k<starNo;k++)
        {
            printf("*");
            }
        printf("\n");
     }

     for(l=0;l<3;l++)
     {
            for(m=0;m<(rows*2+1)/2;m++)
            {
            printf(" ");
        }
        printf("*\n");
 }
return 0;
 }
```

Q2. Write a program that reads in a month number and outputs the month name.

```
#include <stdio.h>
int main()
{
   int monno;
   printf("Input Month No : ");
```

```
scanf("%d",&monno);
switch(monno)
{
        case 1:
                printf("January\n");
                break;
        case 2:
                printf("February\n");
                break;
        case 3:
                printf("March\n");
                break;
        case 4:
                printf("April\n");
                break;
        case 5:
                printf("May\n");
                break;
        case 6:
                printf("June\n");
                break;
        case 7:
                printf("July\n");
                break;
        case 8:
                printf("August\n");
                break;
        case 9:
                printf("September\n");
                break;
        case 10:
                printf("October\n");
                break;
        case 11:
                printf("November\n");
                break;
        case 12:
                printf("December\n");
                break;
        default:
                printf("invalid Month number \n");
                break;
  }
```

```
    return 0;
}
```

Q3. Write a program to demonstrate the use of various input statements like getchar(), getch(),scanf()

```
#include<stdio.h>
#include<string.h>
int main()
{
char ch,nm;
char name[20];
int age,i=0;
printf("would you like to enter your details");
printf("enter Y/N");
ch=getch();

if(ch=='Y')
   {
     printf("enter your name");
     while(nm!='\n')
     {
        nm=getchar();
        name[i]=nm;
        i++;
     }
     name[i]='\0';
     printf("enter your age");
     scanf("%d",&age);

     printf("your name is");
     for(i=0;i<strlen(name);i++)
        printf("%c",name[i]);
     printf("your age is");
     printf("%d",age);

   }
   else
   {
      printf("well,you have not provided the details" );

   }
```

```
return 0;
}
```

Q4. Write a program to demonstrate the overflow and underflow of various datatype.

```
/* overflow and underflow in integer datatype*/

#include <stdio.h>
   int main(void)

     {
          int l, x;
          l = 0x40000000;
          printf("l = %d (0x%x)\n", l, l);
/*addition causing overflow*/
          x = l + 0xc0000000;
          printf("l + 0xc0000000 = %d (0x%x)\n", x, x);

/* multiplication causing overflow*/
          x = l * 0x4;
          printf("l * 0x4 = %d (0x%x)\n", x, x);
/* subtraction causing underflow*/
          x = l - 0xffffffff;
          printf("l - 0xffffffff = %d (0x%x)\n", x, x);
          return 0;
   }
```

Q5. Write a program to demonstrate the precedence of various operators

```
/* precedence of arithmetic and unary operators*/

int main( )

{

int ans, val=4;val = val + 1 ;


printf("ans=%d val=%d\n",ans,val);

val++ ; ++val ;

printf("ans=%d val=%d\n",ans,val);

ans = 2 * val++ ;

printf("ans=%d val=%d\n",ans,val);
```

```
val--;
--val;
val;
val;
printf("ans=%d val=%d\n",ans,val);
ans=--val*2;
printf("ans%dval%d\n", ans, val);
printf("ans=%dval=%d\n",ans,val);
ans = val-- / 3 ;
printf("ans=%d val=%d\n",ans,val);
return 0;
}
```

Assignments to be done by students in Lab

1. Draw a pyramid using '*'
2. Demonstrate the overflow and underflow in char data type
3. Demonstrate the resolution of overflow and underflow
4. Demonstrate the precedence of relational and logical operators

Programming in C/C++

LAB MANUAL

Lab Exercises for Week 2

Q1. Write a program to generate a sequence of numbers in both ascending and descending order.
Q2. Write a program to generate pascals triangle
Q3. Write a program to reverse the digits of a given number.
Q4. Write a program to convert an amount in figures to equivalent amount in words. a. Convert an amount (in millions) to equivalent amount in words b. Convert an amount (in billions) to equivalent amount in words
Q5. Write a program to find sum of all prime numbers between 100 and 500.

Q1. Write a program to generate a sequence of numbers in both ascending and descending order.

```
#include <stdio.h>
 int main() {
      int n, data[100], i, j, temp;

      /* get the number of entries */
      printf("Enter your input for n:");
      scanf("%d", &n);

      /* get the input sequence */
      for (i = 0; i < n; i++)
            scanf("%d", &data[i]);

      /* sort the given sequence in ascending order */
      for (i = 0; i < n-1; i++) {
            for (j = i + 1; j < n; j++) {
                  if (data[i] > data[j]) {
                        temp = data[i];
                        data[i] = data[j];
                        data[j] = temp;
                  }
            }
      }

      /* sequence in ascending order */
      printf("Ascending Order:\n");
      for (i = 0; i < n; i++)
            printf("%d\n", data[i]);

      /* sequence in descending order */
      printf("\nDescending Order:\n");
      for (i = n-1; i >= 0; i--)
            printf("%d\n", data[i]);

      return 0;
 }
```

Q2. Write a program to generate pascals triangle

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

```
long factorial(int);
int main()
{
  int i, n, c;

  printf("Enter the number of rows you wish to see in pascal triangle\n");
  scanf("%d",&n);

  for (i = 0; i < n; i++)
  {
    for (c = 0; c <= (n - i - 2); c++)
      printf(" ");

    for (c = 0 ; c <= i; c++)
      printf("%ld ",factorial(i)/(factorial(c)*factorial(i-c)));

    printf("\n");
  }

  return 0;
}

long factorial(int n)
{
  int c;
  long result = 1;

  for (c = 1; c <= n; c++)
      result = result*c;

  return result;
}
```

Q3. Write a program to reverse the digits of a given number.

```
#include <stdio.h>

int main()
{
  int n, reverse = 0;

  printf("Enter a number to reverse\n");
  scanf("%d", &n);
```

```
    while (n != 0)
    {
      reverse = reverse * 10;
      reverse = reverse + n%10;
      n = n/10;
    }

    printf("Reverse of entered number is = %d\n", reverse);

    return 0;
  }
```

Q4. Write a program to convert an amount in figures to equivalent amount in words.

```
/*Converts an amount (in millions) to equivalent amount in words*/

#include<stdio.h>

void pw(long,char[]);
char *one[]={" "," one"," two"," three"," four"," five"," six"," seven",
"eight"," Nine"," ten"," eleven"," twelve"," thirteen"," fourteen",
"fifteen"," sixteen"," seventeen"," eighteen"," nineteen"};
char *ten[]={" "," "," twenty"," thirty"," forty"," fifty"," sixty",
"seventy"," eighty"," ninety"};

int main()
{
 long n;
 printf("Enter any 9 digit no: ");
 scanf("%9ld",&n);
 if(n<=0)
            printf("Enter numbers greater than 0");
 else
 {
              pw((n/10000000),"crore");
              pw(((n/100000)%100),"lakh");
              pw(((n/1000)%100),"thousand");
              pw(((n/100)%10),"hundred");
              pw((n%100)," ");
 }
 return 0;
}
```

```
void pw(long n,char ch[])
{
 (n>19)?printf("%s %s ",ten[n/10],one[n%10]):printf("%s ",one[n]);
 if(n)printf("%s ",ch);
}
```

Q5. Write a program to find sum of all prime numbers between 100 and 500.

```
#include<stdio.h>
int main()
{
int i,j,sum=0;
printf("the prime numbers are:\n");
for(i=100;i<=500;i++)
{
for(j=2;j<i;j++)
{
if(i%j==0)
break;
}

if(i==j)
printf("%d\t",i);
sum=sum+i;
}

printf("sum of prime numbers between 100 and 500 is %d",sum);
return 0;

}
```

Assignments to be done by students in Lab

1. Convert an amount (in billions) to equivalent amount in words

Programming in C/C++

LAB MANUAL

Lab Exercises for Week 3

Q1. Create a one dimensional array of characters and store a string inside it by reading from standard input.
Q2. Write a program to input 20 arbitrary numbers in one dimensional array. Calculate the frequency of each number. Print the number and its frequency in a tabular form.
Q3. Write a C function to remove duplicates from an ordered array.
Q4. Write a program which will arrange the positive and negative numbers in one dimensional array in such a way that all negative numbers should come first and then all the positive numbers will come without changing the original sequence of numbers.
Q5. Write a program to perform following operations on a 2D array a. Addition b. Multiplication c. Transpose

Q1. Create a one dimensional array of characters and store a string inside it by reading from standard input.

```
#include<stdio.h>
int main()
{
char msg[50], ch;
int i=0;
printf("character strings\n");
printf("Type the characters terminated by a Return or Enter\n");
while((ch=getchar())!='\n')
msg[i++]=ch;
msg[i]='\0';
i=0;
while(msg[i]!='\0')
putchar(msg[i++]);
printf("\n");
return 0;
}
```

Q2. Write a program to input 20 arbitrary numbers in one dimensional array. Calculate the frequency of each number. Print the number and its frequency in a tabular form.

```
#include <stdio.h>

int main()
{
    int arr[100], freq[100];
    int size, i, j, count;
```

```
    /* Input size of array */
    printf("Enter size of array: ");
    scanf("%d", &size);

    /* Input elements in array */
    printf("Enter elements in array: ");
    for(i=0; i<size; i++)
    {
        scanf("%d", &arr[i]);

        /* Initially initialize frequencies to -1 */
        freq[i] = -1;
    }



    for(i=0; i<size; i++)
    {
        count = 1;
        for(j=i+1; j<size; j++)
        {
            /* If duplicate element is found */
            if(arr[i]==arr[j])
            {
                count++;

                /* Make sure not to count frequency of same element
again */
                freq[j] = 0;
            }
        }
```

```
        /* If frequency of current element is not counted */
        if(freq[i] != 0)
        {
            freq[i] = count;
        }
    }

    /*
     * Print frequency of each element
     */
    printf("\nFrequency of all elements of array : \n");
    for(i=0; i<size; i++)
    {
        if(freq[i] != 0)
        {
            printf("%d occurs %d times\n", arr[i], freq[i]);
        }
    }

    return 0;
}
```

Q3. Write a C function to remove duplicates from an ordered array.

```
#include <stdio.h>

    int main()
    {
      int n, a[100], b[100], count = 0, c, d;

      printf("Enter number of elements in array\n");
```

```
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &a[c]);

  for (c = 0; c < n; c++)
  {
    for (d = 0; d < count; d++)
    {
      if(a[c] == b[d])
        break;
    }
    if (d == count)
    {
      b[count] = a[c];
      count++;
    }
  }

  printf("Array obtained after removing duplicate elements:\n");

  for (c = 0; c < count; c++)
    printf("%d\n", b[c]);

  return 0;
}
```

Q4. Write a program which will arrange the positive and negative numbers in one dimensional array in such a way that all positive numbers should come first and then all the negative numbers will come without changing the original sequence of numbers.

```
#include<stdio.h>


int main()

{

int pos[20], count=0,neg[20],index=0;

int i, j, a[20];


for(i=0;i<6;i++)

    scanf("%d",&a[i]);

for(i=0;i<6;i++)

{

  if (a[i]>=0)

  {

     pos[index]=a[i];

     index++;

  }

  else

  {

     neg[count]=a[i];

     count++;

  }

}

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

a[i]=pos[i];

for(i=index, j=0;i<6;i++)

{a[i]=neg[j];

j++;
```

```
}
for(i=0;i<6;i++)
    printf("%d",a[i]);
return 0;
}
```

Q5. Write a program to perform following operations on a 2D array

a. Addition

```
# include<stdio.h>

int main()
{
    int a[2][2], b[2][2],s[2][2],i, j;
    printf("enter the elements of first matrix\n");
  for(i=0;i<=1;i++)
   {
      for(j=0;j<=1;j++)
       {
         scanf("%d",&a[i][j]);
       }
   }

   printf("enter the elements of second matrix\n");
  for(i=0;i<=1;i++)
   {
      for(j=0; j<=1;j++)
       {
         scanf("%d",&b[i][j]);
       }
   }
```

```
   for(i=0;i<=1;i++)
   {
      for(j=0; j<=1;j++)
       {
         s[i][j]=a[i][j]+b[i][j];
       }
    }

printf("the sum of two matrices is \n");

for (i=0;i<=1;i++)
 {
   for (j=0;j<=1;j++)
   {
      printf("%d",s[i][j]);
   }
   printf("\n");
 }
}
```

b. Multiplication

```
# include<stdio.h>

int main()
{
    int a[10][10], b[10][10],s[10][10];
    int m, n, l,p, i, j, k;
    printf("enter the row of first matrix(<=10):");
    scanf("%d",&m);
```

```
  printf("enter the column of first matrix(<=10):");
  scanf("%d",&n);

  printf("enter the row of second matrix(<=10):");
  scanf("%d",&l);

  printf("enter the row of second matrix(<=10):");
  scanf("%d",&p);



  printf("enter the elements of first matrix\n");
for(i=0;i<=m-1;i++)
 {
    for(j=0;j<=n-1;j++)
     {
       scanf("%d",&a[i][j]);
     }
 }

 printf("enter the elements of second matrix\n");
for(i=0;i<=l-1;i++)
 {
    for(j=0; j<=p-1;j++)
     {
       scanf("%d",&b[i][j]);
     }
 }



 if(n!=l)
```

```
 printf("multiplication not possible");

else{

 for(i=0;i<=m-1;i++)
{
   for(j=0; j<=p-1;j++)
    {
      s[i][j]=0;
      for(k=0;k<=n-1;k++)
      {
          s[i][j]=s[i][j]+a[i][k]*b[k][j];
      }
    }
  }
}

printf("matrix multiplication is \n");

for(i=0;i<=m-1;i++)
{
   for(j=0; j<=p-1;j++)
    {
      printf("%d\n", s[i][j]);
    }
    printf("\n");
 }
}
```

c. Transpose

```
# include<stdio.h>

int main()
{
    int a[3][3], b[3][3],i, j,m,n;

    printf("enter the number of rows of matrix\n");
    scanf("%d",&n);

    printf("enter the number of columns of matrix\n");
    scanf("%d",&m);

    printf("enter the elements of first matrix\n");

    for(i=0;i<=n-1;i++)
   {
      for(j=0;j<=m-1;j++)
       {
         scanf("%d",&a[i][j]);
         b[j][i]=a[i][j];
       }
   }
   printf("Transpose of matrix is:\n");
  for(i=0;i<=m-1;i++)
   {
      for(j=0; j<=n-1;j++)
       {
         printf("%d",b[i][j]);
```

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

Lab Exercises for Week 4

Q1. Write a program to find the GCD and LCM of two numbers
Q2. Implement a swap () function which exchanges the values of two integers. Call the function from the main to test the function with different values.
Q3. Write a program to remove duplicates from an ordered array
Q4. Write a function to generate the Fibonacci series using recursion.
Q5. Write a recursive function that adds first 'n' natural numbers.
Q6. Write a recursive function that finds factorial of a number
Q7. Write a program to demonstrate the use of recursion in Tower of Hanoi problem

Q1. Write a program to find the GCD and LCM of two numbers

```
#include <stdio.h>

 void main()
{
int num1, num2, gcd, lcm, remainder, numerator, denominator;

printf("Enter two numbers\n");
scanf("%d %d", &num1, &num2);
if (num1 > num2)
{
numerator = num1;
denominator = num2;
}
else
{
numerator = num2;
denominator = num1;
}
 remainder = numerator % denominator;
while (remainder != 0)
{
numerator = denominator;
denominator = remainder;
remainder = numerator % denominator;
}
gcd = denominator;
```

```
 lcm = num1 * num2 / gcd;
  printf("GCD of %d and %d = %d\n", num1, num2, gcd);
 printf("LCM of %d and %d = %d\n", num1, num2, lcm);
 }
```

Q2. Implement a swap () function which exchanges the values of two integers. Call the function from the main to test the function with different values.

```
#include<stdio.h>
void swap (int *, int *);

int main()
{
int x, y;
printf("enter the values of x and y");
scanf("%d %d",&x,&y);

printf("before exchange: x=%d y=%d\n\n",x,y);
swap(&x,&y);

printf("after exchange: x=%d y=%d\n\n",x,y);
return 0;
}
void swap(int *a, int *b)
{
int t;
t=*a;
*a=*b;
*b=t;
}
```

Q3. Write a program to remove duplicates from an ordered array

```
#include <stdio.h>

int main()
{
  int n, a[100], b[100], count = 0, c, d;

  printf("Enter number of elements in array\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &a[c]);

  for (c = 0; c < n; c++)
  {
    for (d = 0; d < count; d++)
    {
      if(a[c] == b[d])
        break;
    }
    if (d == count)
    {
      b[count] = a[c];
      count++;
    }
  }

  printf("Array obtained after removing duplicate elements:\n");
```

```
      for (c = 0; c < count; c++)
        printf("%d\n", b[c]);

      return 0;
    }
```

Q4. Write a function to generate the Fibonacci series using recursion.

```
# include<stdio.h>

void fseries(int);

int main()
{
int limit, f0=0, f1=1;
printf("enter the limit of fibonacci series");
scanf("%d", &limit);

if(limit>2)
{
printf("%d\n%d",f0,f1);
fseries(limit-2);
}

else if(limit==2)
{
printf("\n%d\n%d", f0,f1);

}
else if(limit==1)
```

```
printf("%d",f1);
else
printf("series not possible");
return 0;
}

void fseries(int p)
{
int fib;
static int f0=0,f1=1;

if(p==0)
{
printf("\n the series ends here");
}
else
{
fib=f0+f1;
f0=f1;
f1=fib;
printf("\n %d", fib);
fseries(p-1);
}
}
```

Q5. Write a recursive function that adds first ‘n’ natural numbers.

```
# include<stdio.h>

int sum(int a);
```

```
void main()
{
int x,y;
printf("enter any positive int number");
scanf("%d", &y);

printf("the sum of first %d integers=%d\n", y, sum(y));
}

int sum(int y)
{
 if (y==0)
   return 0;

else
return(y+sum(y-1));
}
```

Q6. Write a recursive function that finds factorial of a number

```
# include<stdio.h>
int fact(int);
int main()
{
int n, result;
printf("enter the number");
scanf("%d", &n);
result=fact(n);

printf("\n the factorial of %d is %d",n,result);
return 0;
```

```
}

int fact(int x)
{
if(x==1)
return(x);

else
return(x*fact(x-1));
}
```

Q7. Write a program to demonstrate the use of recursion in Tower of Hanoi problem

```
# include<stdio.h>
void hanoi_tower(char,char,char,int);

int main()
{
int n;
printf("\n input the number of disc");
scanf("%d",&n);

printf("\n Tower of Hanoi for %d DISC",n);
hanoi_tower('X','Y','Z',n);

}

void hanoi_tower(char peg1,char peg2,char peg3, int n)
{
    if(n<=0)
```

```
printf("\n illegal entry");

if(n==1)
printf("\n Move Disk from %c to %c", peg1,peg3);

else
{
hanoi_tower(peg1,peg3,peg2,n-1);
hanoi_tower(peg1,peg2,peg3,1);
hanoi_tower(peg2,peg1,peg3,n-1);
}
}
```

Lab Exercises for Week 5

Q1. Implement a program which uses multiple files for holding multiple functions which are compiled separately, linked together and called by main(). Use static and extern variables in these files.
Q2. Implement a function which receives a pointer to a student struct and sets the value of its fields.
Q3. Write a program that takes five arguments on command line, opens a file and writes one argument per line in that file and closes the file
Q4. Write a program which creates Student (struct) objects using malloc and stores their pointers in an array. It must free the objects after printing their contents.
Q5 Write a function char* stuff (char *s1, char* s2, int sp, int rp) to suff string s2 in string s1 at position sp, replacing rp number of characters (rp may be zer0).
Q6. Write a program to input name, address and telephone number of 'n' persons (n<=20). Sort according to the name as primary key and address as the secondary key. Print the sorted telephone directory.

Q2. Implement a function which receives a pointer to a student struct and sets the value of its fields.

```
#include<stdio.h>
struct student
{
char name[20];
int marks;
};

void print(struct student *ptr);

int main()
{
struct student s1;
print(&s1);
printf("%s %d", s1.name,s1.marks);
return 0;
}

void print(struct student *ptr)
{
printf("enter the name");
scanf("%s",ptr->name);
printf("enter the marks");
scanf("%d",&ptr->marks);
}
```

Q3. Write a program that takes five arguments on command line, opens a file and writes one argument per line in that file and closes the file

```
#include <stdio.h>

#include <stdlib.h>


 /*for exit macro*/

int main(int argc, char **argv)

{

      int e = EXIT_SUCCESS;

      int i=0;

      for ( i = 1; i < argc; i++)

      printf("Argument %d: [%s]\n", i, argv[i]);

      /*open a file*/

      char *path="output.txt";

      FILE *file = fopen(path, "w");

if (!file)

      {

      perror(path);

      return EXIT_FAILURE;

      }

for ( i = 1; i < argc; i++)

      fprintf(file,"%s\n",argv[i]);

/* Close file */

if (fclose(file))

      {

      perror(path);

      return EXIT_FAILURE;

      }
```

```
return e;
}
```

Q4. Write a program which creates Student (struct) objects using malloc and stores their pointers in an array. It must free the objects after printing their contents.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define RECORDS 6


int main()
{
struct student
      {
       char name[25];
      };

struct student *bonds[RECORDS];
int index=0;
int count=0;

while(index < RECORDS)
      {
      bonds[index] = (struct student *)malloc(sizeof(struct student));
      strcpy(bonds[index]->name,"XYZ");
      index++;
      }
while(count<index)
      {
      printf("%s",bonds[count]->name);
```

```
      count++;
      }
free(bonds);
return 0;
}
```

Q5. Write a function char* stuff (char *s1, char* s2, int sp, int rp) to suff string s2 in string s1 at position sp, replacing rp number of characters (rp may be zer0).

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

char * stuff(char *s1,char *s2,int sp, int rp)
{
      int i=0;
while ((rp!=0)&& (s1[sp]!='\0'))
      {
      s1[sp] =s2[i];
      sp++;
      i++;
      rp--;
       }
return s1;
}

/*WORKING PROGRAM*/
int main()
{
char t1[]="helicopter";
char t2[]="XYZ";
char *t3;
t3=stuff(t1,t2,3,4);
puts(t3);
return 0;
```

```
}
```

Q6. Write a program to input name, address and telephone number of 'n' persons (n<=20). Sort according to the name as primary key and address as the secondary key. Print the sorted telephone directory.

```
#include <stdio.h>
#define SIZE 3

int main ()
{
struct jb
      {
      char name[25];
      char address[50];
      int telephone;
      };

struct jb bonds[20];
    /*Array of structures*/
struct jb temp;
/*temporary variable*/
/* get the name of the persons*/
int i=0;
int x,a,b;
int phno;
char getName[25];
char getAddress[50];
/*getting ur data */
while(i<SIZE)
      {
      phno=0;
      printf("Enter your Name");
      scanf("%s",&getName);
      printf("Enter your Address");
      scanf("%s",&getAddress);
```

```
      strcpy(bonds[i].name, getName);
      strcpy(bonds[i].address, getAddress);
      printf("Enter your phone");
      scanf("%d",&phno);
      bonds[i].telephone=phno;
      i++;
      }
printf("initial array display\n");
for(x=0;x<SIZE;x++)
printf("%s\n",bonds[x].name);


/*sorting on the basis of name*/
for(a=0;a<SIZE-1;a++)
for(b=a+1;b<SIZE;b++)
      {
            x = 0;
while(bonds[a].name[x])
{
      if((bonds[a].name[x]) > (bonds[b].name[x]))
                  {
       temp = bonds[a];
      bonds[a] = bonds[b];
      bonds[b] = temp;
      break;
      }

else if((bonds[a].name[x]) < (bonds[b].name[x]))
break;
else
x++;
/* if the first character is matching */
   }
 }
printf("modified array:");
```

```
for(x=0;x<SIZE;x++)
      {
      printf("%s\n",bonds[x].name);
      printf("%s\n",bonds[x].address);
      printf("%d\n",bonds[x].telephone);
      }
return (0);
}
```

Lab Assignment

Implement a program which uses multiple files for holding multiple functions which are compiled separately, linked together and called by main(). Use static and extern variables in these files.

Lab Exercises for Week 6

Q1.a. Write a program to find the number of words in a sentence. b. Write a program to find the number of occurrences of particular word in a sentence.
Q2. Write a program to concatenate two strings without using the inbuilt function.
Q3. Write a program to check if two strings are same or not.
Q4. Write a program to check whether a string is a palindrome or not.
Q5. Write a program to find the number of vowels and consonants in a sentence
Q6. Write a program that reverse the contents of a string.

Q1. a. Write a program to find the number of words in a sentence.

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

int main()
{
 char strs[100], countw=0, strw[15], i;
 printf("input sentence : ");
 gets(strs);
 int len=strlen(strs);
 for(i=0; i<len; i++)
 {
 if(strs[i]==' ')
 {
 countw++;
 }
 }
 printf("Total number of words in the sentence is %d",countw+1);

 return 0;
}
```

Q2. Write a program to concatenate two strings without using the inbuilt function.

```
    #include <stdio.h>
    int main()
    {
        char s1[100], s2[100], i, j;
```

```
        printf("Enter first string: ");
        scanf("%s", s1);
        printf("Enter second string: ");
        scanf("%s", s2);
        // calculate the length of string s1
        // and store it in i
        for(i = 0; s1[i] != '\0'; ++i);
        for(j = 0; s2[j] != '\0'; ++j, ++i)
        {
            s1[i] = s2[j];
        }
        s1[i] = '\0';
        printf("After concatenation: %s", s1);
        return 0;
    }
```

Q3. Write a program to check if two strings are same or not.

```
#include<stdio.h>

int main()
{
char string1[50];
char string2[50];
int i, j, flag;
printf("enter the string\n");
gets(string1);
printf("enter second string\n");
gets(string2);
i=0;
j=0;
flag=0;
```

```
while(string1[i]!='\0')
i++;
while(string2[j]!='\0')
j++;
if(i!=j)
flag=1;
else
{
i=0;
j=0;
while((string1[i]!='\0')&&(string2[j]!='\0'))
{
if (string1[i]!=string2[j])
{
flag=1;
break;
}
j++;
i++;
}
}
 if(flag==0)
   printf("\n strings are equal");
else
   printf("\n strings are not equal");
   return 0;
}
```

Q4. Write a program to check whether a string is a palindrome or not.

```
#include<stdio.h>
#include<string.h>
void isPalindrome(char str[]);

int main()
{

    char s1[100];
    printf("Enter the string: ");
    scanf("%s", s1);
    isPalindrome(s1);
    return 0;
}

// A function to check if a string str is palindrome
void isPalindrome(char str[])
{
    // Start from leftmost and rightmost corners of str
    int l = 0;
    int h = strlen(str) - 1;

    // Keep comparing characters while they are same
    while (h > l)
    {
        if (str[l++] != str[h--])
        {
            printf("%s is Not Palindrome", str);
            return;
        }
```

```
    }
    printf("%s is palindrome", str);
}
```

Q5. Write a program to find the number of vowels and consonants in a sentence.

```
#include <stdio.h>
#include <string.h>
#define MAX_SIZE 100

int main()
{
    char str[MAX_SIZE];
    int i, len, vowel, consonant;

    /* Input string from user */
    printf("Enter any string: ");
    gets(str);

    vowel = 0;
    consonant = 0;
    len = strlen(str);

    for(i=0; i<len; i++)
    {
        if((str[i]>='a' && str[i]<='z') || (str[i]>='A' &&
str[i]<='Z'))
        {
            /*
             * If the current character(str[i]) is a vowel
```

```
            */
            if(str[i] =='a' || str[i]=='e' || str[i]=='i' ||
str[i]=='o' || str[i]=='u' ||
               str[i] =='A' || str[i]=='E' || str[i]=='I' ||
str[i]=='O' || str[i]=='U'  )
                vowel++;
            else
                consonant++;
        }
    }

    printf("Total number of vowel = %d\n", vowel);
    printf("Total number of consonant = %d\n", consonant);

    return 0;
    }
```

Q6. Write a program that reverse the contents of a string.

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

int main() {
   char str[100], temp;
   int i, j = 0;

   printf("\nEnter the string :");
   gets(str);

   i = 0;
   j = strlen(str) - 1;
```

```
   while (i < j) {
      temp = str[i];
      str[i] = str[j];
      str[j] = temp;
      i++;
      j--;
   }
printf("\nReverse string is :%s", str);
   return (0);
}
```

Assignments

Write a program to find the number of occurrences of particular word in a sentence.

Lab Exercises for Week 7

Q1. Write a program to demonstrate the array indexing using pointers.
Q2. Write a program to pass a pointer to a structure as a parameter to a function and return back a pointer to structure to the calling function after modifying the members of structure.
Q3. Write a program to demonstrate the use of pointer to a pointer.
Q4. Write a program to demonstrate the use of pointer to a function.
Q5. Write a program to demonstrate the swapping the fields of two structures using pointers

Q1. Write a program to demonstrate the array indexing using pointers.

```
#include<stdio.h>

int main()
{
int *p, sum=0, i;
int x[5]={5,9,6,3,7};
i=0;
// initializing with base address of x
p=x;
printf("Element value address\n\n");

while(i<5)
{
printf("x[%d] %d %u\n",i, *p,p);
sum=sum+*p; /* accessing array element*/
i++; p++;

}
printf("\n Sum=%d\n", sum);
printf("\n &x[0]=%u\n",&x[0]);
printf("\n p=%u\n",p);
}
```

Q2. Write a program to pass a pointer to a structure as a parameter to a function and return back a pointer to structure to the calling function after modifying the members of structure.

```
#include<stdio.h>


struct student
{
char name[20];
int marks;
};

struct student * print(struct student *ptr);

int main()
{
struct student s1,*s2;

printf("enter the name");
scanf("%s",s1.name);
printf("enter the marks");
scanf("%d",s1.marks);

s2=print(&s1);
printf("contents after modifying\n");
printf("%s %d", s2->name,s2->marks);


return 0;
}

struct student * print(struct student *ptr)
{
```

```
printf("enter the name");
scanf("%s",ptr->name);
printf("enter the marks");
scanf("%d",&ptr->marks);
return (ptr);
}
```

Q3. Write a program to demonstrate the use of pointer to a pointer.

```
#include<stdio.h>

int main()
{
int x, *p, **q;
x=10;
p=&x;
q=&p;
// print the value of x
printf("%d", **q);
return 0;
}
```

Q4. Write a program to demonstrate the use of pointer to a function.

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

void check(char *a, char *b, int (*cmp)(const char *, const char *));

int main()
{
```

```
char s1[80],s2[80];
int (*p) (const char *, const char *);
p=strcmp;

gets(s1);
gets(s2);

check(s1,s2,p);

return 0;
}

void check(char *a, char *b, int (*cmp)(const char *, const char *))
{
printf("testing for equality\n");
if(!(*cmp)(a,b))
printf("Equal");

else
printf("Not Equal");

}
```

Assignment

Q1. Write a program to demonstrate the swapping the fields of two structures using pointers

Lab Exercises for Week 8

Q1. Write a program in C++ to define class complex having two data members viz real and imaginary part.
Q2. Write a program in C++ to define class Person having multiple data members for storing the different details of person e.g. name, age, address, height.
Q3. Write a program to instantiate the objects of class person and class complex
Q4. Write a C++ program to add member function that displays the contents of class person and class complex.
Q5. Write a C++ program to demonstrate the use of scope resolution operator.

Q1. Write a program in C++ to define class complex having two data members viz real and imaginary part.

```
#include <iostream>
using namespace std;

class complex
{
private:
float real;
float imaginary;

public:
void getdata()
{

real=3.5;
imaginary=7.5;
}

void putdata()
{
cout<<real<<" +i"<<imaginary;

}
};

int main()
{
complex c;
c. getdata();
```

```
c. putdata();
return 0;
}
```

Q2. Write a program in C++ to define class Person having multiple data members for storing the different details of person e.g. name, age, address, height.

```
#include <iostream>
using namespace std;

class person{

private:
char name[20];
char address[20];
int age;
float height;

public:

void getdata()
{
cout<<"enter name"<<endl;
cin>>name;
cout<<"enter address"<<endl;
cin>>address;

cout<<"enter age"<<endl;
cin>>age;
```

```
cout<<"enter height"<<endl;
cin>>height;

}

void putdata()
{
cout<<"name is: "<<name<<endl;
cout<<"address is: "<<address<<endl;
cout<<"age is: "<<age<<endl;
cout<<"height is: "<<height<<endl;

}
};


int main()
{
person p;
p.getdata();
p.putdata();
return 0;
}
```

Q4. Write a C++ program to add member function that displays the contents of class person and class complex.

```
#include <iostream>
using namespace std;

class complex
```

```
{
private:
float real;
float imaginary;

public:

void get_complex()
{
cout<<" enter the real part"<<endl;
cin>> real;

cout<<"enter the imaginary part"<<endl;
cin>>imaginary;

}

void show_complex()
{
cout<<real<<"+i"<<imaginary;
}
};

class person{

private:
char name[20];
char address[20];
int age;
float height;
```

```
public:

void getdata()
{
cout<<"enter name"<<endl;
cin>>name;
cout<<"enter address"<<endl;
cin>>address;

cout<<"enter age"<<endl;
cin>>age;

cout<<"enter height"<<endl;
cin>>height;

}

void putdata()
{
cout<<"name is: "<<name<<endl;
cout<<"address is: "<<address<<endl;
cout<<"age is: "<<age<<endl;
cout<<"height is: "<<height<<endl;

}
};


int main()
```

```
{
complex c1;
c1.get_complex();
c1.show_complex();

person p1;
p1.getdata();
p1.putdata();

return 0;

}
```

Q5. Write a C++ program to demonstrate the use of scope resolution operator.

```
#include <iostream>
#include <string.h>
using namespace std;


class student
{
private:
char name[20];
int marks;

public:

void put();
void get(char *n, int a);
```

```
};

void student::get(char *n, int a)
{
strcpy(name,n);
marks=a;
}


void student::put()
{
cout<< "name is:"<<name<<endl;
cout<< " marks:"<<marks<<endl;
}


int main()
{
 student s1;
 s1.get("XYZ", 56);
 s1.put();
 return 0;
}
```

Assignment

Q. Write a program to instantiate the objects of class person and class complex

Lab Exercises for Week 9

Q1. Write a program in C++ which creates objects of Student class using default, overloaded and copy constructors.
Q2. Write a program to demonstrate the use of different access specifiers.
Q3. Write a C++ program to demonstrate the use of inline, friend and this keyword.
Q4. Write a C++ program to show the use of destructors.
Q5. Write a program in C++ to demonstrate the use of function overloading.

Q1. Write a program in C++ which creates objects of Student class using default, overloaded and copy constructors.

```
#include <iostream>
using namespace std;
#include <string.h>

class student{

private:
char name[20];
int age;

public:
student(){};
student(char *n)
{
strcpy(name,n);
age=0;
}

student(char *n, int a)
{
strcpy(name,n);
age=a;
}

student(student &s)
{
strcpy(name,s.name);
age=s.age;
```

```
}
void show();

};
void student:: show()
{
cout<< "name of student is:"<<name<<endl;
cout<<" age of student is:"<<age<<endl;
}


int main()
{

student s2("XYZ");
student s3("abc",26);
student s4(s3);


s2.show();
s3.show();
s4.show();
return 0;

}
```

Q2. Write a program to demonstrate the use of different access specifiers.

```
    #include<iostream>
using namespace std;
```

```
class sample
{
int m,n;
void display();
public:
void input();
int largest();
};

int sample:: largest()
{
if(m>=n)
return (m);
else
return (n);
}

void sample::input()
{
cout<<"input values of m and n"<<"\n";
cin>>m>>n;
}

void sample:: display()
{
cout<<"largest value="<<largest()<<"\n";
}

int main()
{
```

```
sample A;
int temp;
A.input();
temp=A. largest();
cout<<"largest value="<<temp<<"\n";
//A.display();// objects cant access private members
return 0;
}
```

Q3. Write a C++ program to demonstrate the use of inline, friend and this keyword.

```
#include<iostream>
using namespace std;


class sample
{
int m;
int n;
void display();
public:
friend int sum(sample x);
void set_mn(int m,int n);
void put_mn();
};

void sample::set_mn(int m, int n)
{
this->m=m;
this->n=n;
```

```
}
inline void sample::put_mn()
{
cout<<"m:"<<m<<endl<<"n:"<<n<<endl;
}
int sum(sample x)
{
return x.m+x.n;
}

int main()
{
sample s1,s2;
s1.set_mn(7,8);
s1.put_mn();
s2.set_mn(56,8);
s2.put_mn();
cout<<sum(s1);
return 0;
}
```

Q4. Write a C++ program to show the use of destructors.

```
# include<iostream>
# include<string.h>
using namespace std;
int count=0;

class student
{
```

```
int roll;
int age;
char name[20];

public:
student(int r, int a, char *n);
~student();
void get_details();
};


student::student(int r, int a, char *z)
{
count++;
cout<<"No. of objects created "<<count<<endl;
age=r;
roll=a;
strcpy(name,z);
}

student::~student()
{
cout<<"No. of objects destroyed "<<count<<endl;
count--;
}

void student:: get_details()
{
cout<<"name  "<<name<<endl;
cout<<"age   "<<age<<endl;
```

```
cout<<"roll  "<<roll<<endl;

}

int main()
{
student s1(1, 24,"XYZ");
s1.get_details();
student s2(2, 25, "ABC");
s2.get_details();

}
```

Q5. Write a program in C++ to demonstrate the use of function overloading.

```
#include<iostream>
using namespace std;


class sample
{
public:
int sum(int m, int n);
double sum(double m,double n);
int sum(int m);
};

int sample::sum(int m, int n)
{
```

```
return m+n;
}

double sample::sum(double m,double n)
{
return m+n;
}

int sample::sum(int m)
{
return m+2;
}


int main()
{
sample s1;
cout<<s1.sum(5)<<endl;
cout<<s1.sum(5.6,7.8)<<endl;
cout<<s1.sum(4,5)<<endl;

return 0;
}
```

Lab Exercises for Week 10

Q1. Write a C++ program to overload the '+' operator so that it can add two matrices.
Q2. Write a C++ program to overload the assignment operator.
Q3. a. Write a C++ program to overload comparison operator == b. Write a C++ program to overload !=operator
Q4. Write C++ program to overload the unary operator

Q1. Write a C++ program to overload the '+' operator so that it can add two matrices.

```
# include<iostream>
using namespace std;

class matrices
{
int a[2][2];
int b[2][2];
int c[2][2];

public:
void get_elements();
matrices operator +(matrices m2);
void display();
};

void matrices::get_elements()
{
cout<<"enter the elements";
for(int i=0;i<2;i++)
{
 for(int j=0;j<2;j++)
   cin>>a[i][j];
}
}

void matrices:: display()
{
for(int i=0;i<2;i++)
```

```
{
 for(int j=0;j<2;j++)
   cout<<a[i][j];
   cout<<endl;
}
}
matrices matrices::operator+(matrices m2)
{
matrices m3;
for(int i=0;i<2;i++)
{
  for(int j=0;j<2;j++)
     m3.a[i][j]=a[i][j]+m2.a[i][j];
}

return(m3);
}



int main()
{
matrices ob1,ob2;
ob1.get_elements();
ob2.get_elements();

ob1=ob1+ob2;

ob1.display();

}
```

Q2. Write a C++ program to overload the assignment operator.

```
# include<iostream>
using namespace std;

class sample{
int x,y;

public:

sample(){}
sample(int i, int j)
{
x=i;
y=j;
}

void show()
{
cout<<"x=  "<<x<<endl;
cout<<"y=  "<<y<<endl;
}

sample operator=(sample op2);
};

sample sample::operator=(sample op2)
{
x=op2.x;
```

```
y=op2.y;

return *this;
}

int main()
{
sample ob1(10,20),ob2(30,40),ob3(50,60);
ob1.show();
ob2.show();
ob3.show();

ob1=ob2=ob3;
ob1.show();
ob2.show();

return 0;

}
```

Q3. a.Write a C++ program to overload comparison operator ==

```
# include<iostream>
using namespace std;

class sample{
int x;

public:

void getdata()
```

```
{
cout<<"enter the value of a"<<endl;
cin>>x;
}
void show()
{
cout<<"x="<<x<<endl;

}
int operator==(sample op);
};

int sample::operator==(sample op)
{
int i;
if(x==op.x)
i=1;

else
i=0;

return i;
}

int main()
{
sample s1,s2;
s1.getdata();
s2.getdata();
```

```
s1.show();
s2.show();

if(s1==s2)
cout<<"s1 and s2 are equal"<<endl;

else
cout<<"s1 and s2 are not equal"<<endl;

return 0;
}
```

Q4. Write C++ program to overload the unary operator

```
# include<iostream>
using namespace std;


class sample{
int x,y;

public:

sample(){}
sample(int i, int j)
{
x=i;
y=j;
}

void show()
```

```
{
cout<<"x=  "<<x<<endl;
cout<<"y=  "<<y<<endl;

}

sample operator++();
};

sample sample::operator++( )
{
x++;
y++;
return *this;
}

int main()
{
sample ob1(3,4),ob2(5,6);

ob1.show();
ob2.show();

++ob1;
ob1.show();

ob2=++ob1;

ob1.show();
ob2.show();
```

```
return 0;

}
```

Assignment

Write a C++ program to overload !=operator

Lab Exercises for Week 11

Q1. Write a program in C++ which creates a multilevel inheritance hierarchy of Person, Employee and Teacher classes and creates instances of each class using new and stores them in an array of Person *
Q2. Write a program in C++ which creates a multiple inheritance hierarchy of Teacher classes derived from bot Person, Employee classes. Each class must implement a Show() member function and utilize scope resolution operator.
Q3. Write a C++ program that demonstrates the concept of function overriding.
Q4. Write a C++ program to show inheritance using different levels.

Q1. Write a program in C++ which creates a multilevel inheritance hierarchy of Person, Employee and Teacher classes and creates instances of each class using new and stores them in an array of Person *

```
#include <iostream>
using namespace std;
#include <string.h>

class person
{
protected:
int age;
char name[50];
public:
void get_P()
{
    cout<<"enter name and age"<<endl;
    cin>>name>>age;

}
void show()
{
cout<<"name: "<<name<<endl;
cout<<"age:  "<<age<<endl;
}
};



class Employee: public person
{
protected:
```

```
float salary;
public:
void get_E()
{
    cout<<"enter the salary"<<endl;
    cin>>salary;
}
void Eshow()
{
cout<<"salary: "<<salary<<endl;
}

};


class Teacher: public Employee
{
protected:
char area[50];
public:
void Tshow()
{
show();
Eshow();
cout<<"research_area: "<<area<<endl;
}
void get_T()
{
    cout<<"enter the research area"<<endl;
    cin>>area;
```

```
}

};

int main()
{

person *p[3];

p[0]=new person();
p[0]->get_P();
p[0]->show();

p[1]=new Employee();

//acess only the members of derived type inherited from base
p[1]->get_P();
p[1]->show();
//get full access of the entire derived class using typecast
((Employee *)p[1])->get_E();
((Employee *)p[1])->Eshow();


p[2]=new Teacher();
p[2]->get_P();
  ((Teacher *)p[2])->get_P();
  ((Teacher *)p[2])->get_E();
((Teacher *)p[2])->get_T();
((Teacher *)p[2])->Tshow();
}
```

Q2. Write a program in C++ which creates a multiple inheritance hierarchy of Teacher classes derived from bot Person, Employee classes. Each class must implement a Show() member function and utilize scope resolution operator.

```
#include <iostream>

using namespace std;

#include <string.h>


class person

{

protected:

int age;

char name[50];

public:

person(int a, char *n)

{

age=a;

strcpy(name,n);

}



void show()

{

cout<<"name: "<<name<<endl;

cout<<"age:  "<<age<<endl;

}

};
```

```
class Employee
{
protected:
float salary;
public:

Employee(int s)
{
salary=s;
}

void show()
{
cout<<"salary: "<<salary<<endl;
}

};



class Teacher: public person, Employee
{
protected:
char area[50];
public:
Teacher(int a, char *n, int s, char *ar): Employee(s), person(a, n)
{
strcpy(area,ar);
}
void show()
{
```

```
person::show();
Employee::show();
cout<<"research_area: "<<area<<endl;
}
};

int main()
{
Teacher T1 (21,"ABC",7879,"Comp");
T1.show();
return 0;
}
```

Q3. Write a C++ program that demonstrates the concept of function overriding.

```
#include<iostream>
using namespace std;

class Base
{
public:
void show()
{
cout<<"base "<<endl;
}
};

class Derived: public Base
{
public:
```

```
void show()
{
cout<<"derived"<<endl;
}
};


int main()
{
Derived d;
d.show();//invokes show()in derived class
d.Base::show();//invokes show() in base class
}
```

Q4. Write a C++ program to show inheritance using different levels.

```
# include<iostream>
using namespace std;

class student
{
protected:
int r_number;
public:

void get_number(int a)
{
r_number=a;
}

void put_number()
```

```
{
cout<<"roll No"<<r_number<<endl;
}
};

class test: public student
{
protected:
float sub1,sub2;
public:
void get_marks(float x, float y)
{
sub1=x;
sub2=y;
}

void put_marks()
{
cout<<"marks obtained: "<<endl;
cout<<"Sub1= "<<sub1<<endl;
   cout<<"Sub2= "<<sub2<<endl;
}
};

class sports
{
protected:

float score;
public:
```

```
void get_score(float s)
{
score=s;
}

void put_score()
{
cout<<"Sports wt: "<<score<<endl;

}
};



class result:public test, public sports
{
float total;

public:
void display();
};

void result::display()
{
total=sub1+sub2+score;
put_number();
put_marks();
put_score();

cout<<"Total Score: "<<total<<endl;
}
```

```
int main()
{
result student_1;
student_1.get_number(123);
student_1.get_marks(25.6,22.0);
student_1.get_score(6.0);
student_1.display();
return 0;

}
```

Lab Exercises for Week 12

Q1. Write a C++ program to demonstrate the concepts of abstract class and inner class.
Q2. Write a C++ program to demonstrate the use of virtual functions and polymorphism.
Q3. Write a C++ program to demonstrate the use of pure virtual functions and virtual destructors.
Q4. Write a C++ program to swap data using function templates.
Q5. Write a C++ program to create a simple calculator which can add, subtract, multiply and divide two numbers using template.

Q1. Write a C++ program to demonstrate the concepts of abstract class and inner class.

```
//Abstract class
# include<iostream>
using namespace std;

class number
{
protected:

int val;

public:
void setval(int i)
{
val=i;
}
virtual void show()=0;
};

class hextype:public number{

public:
void show(){
cout<<hex<<val<<endl;

}
};

class dectype:public number{
```

```
public:
void show()
{
cout<<val<<endl;
}
};

class octtype:public number
{
public:
void show()
{
cout<<oct<<val<<endl;
}
};

int main()
{
dectype d;
hextype h;
octtype o;

d.setval(20);
d.show();

h.setval(20);
h.show();

o.setval(20);
o.show();
```

```
return 0;

}

//inner class
#include <iostream>
#include<string.h>

using namespace std;

class student{
private:
  string school;
  string degree;
public:
    void getedu()
    {

        cout<<"enter name of school or university:";
        cin>>school;

        cout<<"enter the highest degree earned\n";
        cout<<"highschool, bachelors, mastters, phd";
        cin>>degree;

    }

    void putedu()const
    {
```

```
        cout<<"\n school or university:"<<school;
        cout<<"\n highest degree earned:"<<degree;

    }


};

class employee{

private:
    string name;
    unsigned long number;
public:
    void getdata()
    {
        cout<<"\n enter last name:";
        cin>>name;

        cout<<"enter number:";
        cin>>number;

    }
    void putdata()const
    {
        cout<<"\n Name:"<<name;
        cout<<"\n Number:"<<number;

    }
```

```
};
    class manager
    {

    private:
        string title;
        double dues;
        employee emp;
        student stu;

    public:
        void getdata()
        {
            emp.getdata();
            cout<<"enter title:";
            cin>>title;
            cout<<"enter golf club dues:";
            cin>>dues;
            stu.getedu();
        }
        void putdata()const
        {
        emp.putdata();
        cout<<"\n title:"<<title;
        cout<<"\n golf club dues:"<<dues;
        stu.putedu();

        }
```

```
  };


class scientist
{

private:
  int pubs;
  employee emp;
  student stu;
  public:
  void getdata()
  {

      emp.getdata();
      cout<<"enter number of pubs:";
      cin>>pubs;

      stu.getedu();
  }
  void putdata()const
  {
  emp.putdata();
  cout<<"\n number of publications:"<<pubs;
  stu.putedu();

  }

};
class laborer
```

```
  {

  private:
    employee emp;
    public:
    void getdata()
    {emp.getdata();
    }
    void putdata()const{
    emp.putdata();
    }
  };


int main()
{

    manager m1;
    scientist s1,s2;
    laborer l1;

    cout<<endl;
    cout<<"\n Enter data for manager 1";
    m1.getdata();

    cout<<"\n Enter data for scientist 1";
    s1.getdata();

     cout<<"\n Enter data for scientist 2";
    s2.getdata();
```

```
     cout<<"\n Enter data for laborer 1";
    l1.getdata();

     cout<<"\n data on manager1";
    m1.putdata();

    cout<<"\n data on scientist1";
    s1.putdata();
    cout<<"\n data on scientist2";
    s2.putdata();
     cout<<"\n data on laborer 1";
    l1.putdata();

    cout<<endl;
    return 0;
}
```

Q2. Write a C++ program to demonstrate the use of virtual functions and polymorphism.

```
# include<iostream>
using namespace std;



class base{
public:
virtual void vfunc()
{
cout<<"This is base class function"<<endl;

}
```

```
};

class derived1: public base
{
public:
void vfunc()
{
cout<<"This is derived1's function"<<endl;

}
};

class derived2: public base{
public:
void vfunc()
{
cout<<"This is derived2's vfunc()"<<endl;
}
};

int main()
{
base *p,b;
derived1 d1;
derived2 d2;

//point to Base
p=&b;
p->vfunc();
```

```
//point to derived1
p=&d1;
p->vfunc();

//point to derived2

p=&d2;
p->vfunc();

return 0;



}
```

Q3. Write a C++ program to demonstrate the use of pure virtual functions and virtual destructors.

```
# include<iostream>
using namespace std;

class number
{
protected:

int val;

public:
void setval(int i)
{
val=i;
}
```

```
virtual void show()=0;
virtual~number()
{
cout<<"number object deleted"<<endl;
}
};

class hextype:public number{

public:
void show(){
cout<<hex<<val<<endl;

}
~hextype()
{
cout<<"hextype object deleted"<<endl;
}
};

class dectype:public number{
public:
void show()
{
cout<<val<<endl;
}

~dectype()
{
cout<<"dectype object deleted"<<endl;
```

```
}
};

class octtype:public number
{
public:
void show()
{
cout<<oct<<val<<endl;
}

 ~octtype()
{
cout<<"octtype object deleted"<<endl;
}
};

int main()
{
    number *ptr;
    dectype d;
    hextype h;
    octtype o;
    ptr=&d;

    ptr->setval(20);
    ptr->show();

    ptr=&h;
    ptr->setval(20);
```

```
    ptr->show();

    ptr=&o;
    ptr->setval(20);
    ptr->show();

    return 0;

}
```

Q4. Write a C++ program to swap data using function templates.

```
#include<iostream>
using namespace std;

template <class T>
void swapargs(T&x, T&y)
{
T temp;
temp=x;
x=y;
y=temp;
}

void fun(int m, int n)
{
cout<<"m and n before swap: "<<m<<" "<<n<<endl;
swapargs(m,n);

cout<<"m and n after swap: "<<m<<" "<<n<<endl;
```

```
}

int main()
{
int i=10, j=20;
fun(i,j);

return 0;

}
```

Assignment

Q. Write a C++ program to create a simple calculator which can add, subtract, multiply and divide two numbers using template.

Lab Exercises for Week 13

Q1. Write a C++ program to demonstrate the concept of exception handling.
Q2. Write a C++ program to create a custom exception.
Q3. Define a class with appropriate data members and member functions which opens an input and output file, checks each one for being open, and then reads name, age, salary of a person from the input file and stores the information in an object, increases the salary by a bonus of 10% and then writes the person object to the output file. It continues until the input stream is no longer needed.

Q1. Write a C++ program to demonstrate the concept of exception handling.

```
# include <iostream>
using namespace std;

int main()
{
cout<<"Start\n";

try //Start a try block
{
cout<<"inside try block\n";
throw 100;//throw an error
cout<<"This will not execute";

}

catch(int i)
{//catch error
cout<<"caught an exception--value is:";
cout<<i<<"\n";

}

cout<<"End";
return 0;



}
```

Q2. Write a C++ program to create a custom exception.

```
#include<iostream>
#include <exception>
class Except: virtual public std::exception {

protected:
 int error_number; ///< Error number
 int error_offset; ///< Error offset
 std::string error_message; ///< Error message

public:
 /** Constructor (C++ STL string, int, int).
 * @param msg The error message
 * @param err_num Error number
 * @param err_off Error offset
 */
 explicit
 Except(const std::string& msg, int err_num, int err_off):
 error_number(err_num),
 error_offset(err_off),
 error_message(msg)
 {}
 /** Destructor.
 * Virtual to allow for subclassing.
 */
 virtual ~Except() throw () {}
 /** Returns a pointer to the (constant) error description.
 * @return A pointer to a const char*. The underlying memory
 * is in possession of the Except object. Callers must
```

```
 * not attempt to free the memory.
 */
 virtual const char* what() const throw () {
 return error_message.c_str();
 }

 /** Returns error number.
 * @return #error_number
 */
 virtual int getErrorNumber() const throw() {
 return error_number;
 }

 /**Returns error offset.
 * @return #error_offset
 */
 virtual int getErrorOffset() const throw() {
  return error_offset;
 }
};
int main()
{

    try {
 throw(Except("Couldn't do what you were expecting", -12, -34));
} catch (const Except& e) {
 std::cout<<e.what()
 <<"\nError number: "<<e.getErrorNumber()
 <<"\nError offset: "<<e.getErrorOffset();
}
```

```
}
```

Q3. Define a class with appropriate data members and member functions which opens an input and output file, checks each one for being open, and then reads name, age, salary of a person from the input file and stores the information in an object, increases the salary by a bonus of 10% and then writes the person object to the output file. It continues until the input stream is no longer needed.

```
#include <iostream>
#include <fstream>
using namespace std;


class Person{
      string name;
      int age;
      float salary;


public: Person(string name, int age, float salary){
      this->name=name;
      this->age=age;
      this->salary=salary;

}


public: void display(){
      cout<<this->name;
      cout<<this->age;
      cout<<this->salary;
```

```
}
public: void updateSalary(){

    this->salary=(this->salary)*1.10;



}
public: void outFile(){
ofstream fout;
fout.open("personList");
fout<<this->name<<endl;
fout<<this->age<<endl;
fout<<this->salary<<endl;
fout.close();
}

};




int main(){
      ifstream is("foo.txt");
      string name;
      int age;
      float salary;
      while (is >> name  >> age >> salary);
      Person obj=Person(name, age, salary);

      obj.display();
```

```
        obj.updateSalary();
          obj.display();
        obj.outFile();

    }
```