1.
```
class soccer
{
   char club[35]; int played, won, points;
   public:
      void datainput()
      {
         cout<<"Name? "; gets(club);
         cout<<"Played Won Points? "; cin>>played>>won>>points;
      }
      void datashow()
      {
         cout<<club<<','<<played<<','<<won<<','<<points<<endl;
      }
      int retwon() { return won; }
      int retpoints() { return points; }
};
```

a) Write a C++ function `void appendrec()`, to append n (n is local variable) objects of the type `soccer` in a binary data file `SOCCER.DAT`. [3]

```
void appendrec()
{
   ofstream fout("SOCCER.DAT", ios::binary|ios::app);
   soccer s; int n;
   cout<<"Number of records to append? "; cin>>n;
   for (int k=0; k<n; k++)
   {
      s.datainput();
      fout.write((char*)&s, sizeof(s));
   }
   fout.close();
}
```

b) Write a C++ function `void showclub()`, to read objects of the type `soccer` stored in a binary data file `SOCCER.DAT` and display the details of all the clubs whose `points` earned is at least 60 and won at most 20 matches. Display an error message if no such record is found.

```
void points60()
{
   ifstream fin("SOCCER.DAT", ios::binary);
   soccer s; int c=0;
   while (fin.read((char*)&s, sizeof(s)))
      if (s.retpoints()>=60 && s.retwon()<=20)
      {
         s.datashow();
         c++;
      }
   fin.close();
   if (c==0)
      cout<<"No such record found\n";
}
```

2. a) Write a C++ function `void countalpha()`, display a text file `PICNIC.TXT` and at the end display number of alphabets present in the text file. [3]

```
void countalpha()
{
   ifstream fin("PICNIC.TXT");
```

```
   char ch; int c=0;
   while (fin.get(ch))
   {
      cout<<ch;
      if (ch>='A' && ch<='Z' || ch>='a' && ch<='z')
         c++;
   }
   fin.close();
   cout<<"Alphabest="<<c<<endl;
}
```

b) Write a C++ function `void countisare()`, display a text file `STORY.TXT` and at the end display number of times "IS" and "ARE" present in the file. Count the words separately and ignore the case. [3]

```
void countisare()
{
   ifstream fin("STORY.TXT");
   char ch;
   while (fin.get(ch))
      cout<<ch;
   fin.close();
   fin.open("STORY.TXT");
   char word[15]; int c1=0, c2=0;
   while (fin>>word)
   {
      if (strcmpi(word, "IS")==0)
         c1++;
      if (strcmpi(word, "ARE")==0)
         c2++;
   }
   fin.close();
   cout<<"Number of IS ="<<c1<<endl;
   cout<<"Number of ARE="<<c2<<endl;
}
```

3. a)

```
struct hdtv
{
   char model[40]; double price; hdtv *next;
};
```

Give the complete class declaration of a linked stacked where each node is of the type `hdtv`. Define constructor, function to insert and function to delete. [5]

```
class stack
{
   hdtv *top;
   public:
      stack() { top=NULL; }
      void push()
      {
         hdtv *p=new hdtv;
         if (p==NULL)
            cout<<"Stack Overflow\n";
         else
         {
            cout<<"Model? "; gets(p->model);
```

```
            cout<<"Price? "; cin>>p->price;
            p->next=top;
            top=p;
         }
      }
      void pop()
      {
         if (top==NULL)
            cout<<"Stack Underflow\n";
         else
         {
            hdtv *p=top;
            cout<<p->model<<" , "<<p->price<<" popped\n";
            top=top->next;
            delete p;
         }
      }
};
```

b)
```
#define MAX 25
struct city
{
   char cityname[30]; double population;
};
```

Give the complete class declaration of an array implemented stacked. Define constructor, function to insert and function to delete. [5]

```
class stack
{
   city arr[MAX]; int top;
   public:
      stack() { top=-1; }
      void push()
      {
         if (top==MAX-1)
            cout<<"Stack Overflow\n";
         else
         {
            top++;
            cout<<"City Name? "; gets(arr[top].cityname);
            cout<<"Population? "; cin>>arr[top].population;
         }
      }
      void pop()
      {
         if (top==-1)
            cout<<"Stack Underflow\n";
         else
         {
            cout<<arr[top].cityname<<" , "<<arr[top].population
                <<" popped\n";
            top--;
         }
      }
};
```

c) Evaluate the postfix expressions given below, showing the content of stack after every step:

i) `TRUE,NOT,FALSE,TRUE,FALSE,OR,FALSE,NOT,TRUE,OR,AND,OR,AND` [2]

Symbol	Stack
TRUE	TRUE
NOT	FALSE
FALSE	FALSE, FALSE
TRUE	FALSE, FALSE, TRUE
FALSE	FALSE, FALSE, TRUE, FALSE
OR	FALSE, FALSE, TRUE
FALSE	FALSE, FALSE, TRUE, FALSE
NOT	FALSE, TRUE, FALSE, TRUE
TRUE	FALSE, TRUE, FALSE, TRUE, TRUE
OR	FALSE, TRUE, FALSE, TRUE
AND	FALSE, TRUE, FALSE
OR	FALSE, FALSE
AND	FALSE

ii) `6, 2, ^, 8, 2, *, -, 10, +, 14, 24, 3, /, -, /` [2]

Symbol	Stack
6	6
2	6, 2
^	36
8	36, 8
2	36, 8, 2
*	36, 16
-	20
10	20, 10
+	30
14	30, 14
24	30, 14, 24
3	30, 14, 24, 3
/	30, 14, 8
-	30, 6
/	5

d) **MAT**[20][30] is a 2D array with a base address 2000. Each element of **MAT** requires 4 bytes. Calculate the address of **MAT**[17][23] when **MAT** is stored as i) row wise ii) col wise [3]

i) **Row** wise: `b+w*(r*NOC+c) = 2000+4*(17*30+23) = 4132`

ii) **Col** wise: `b+w*(r+c*NOR) = 2000+4*(17+23*20) = 3908`

4. a) Give the output of the program given below: [5]

i)
```
#include<iostream.h>
void main()
{
   char *str="VACATION";
   for (char *p=str; *(p+1); p+=2)
      cout<<*p<<*(p+1)<<endl;
}
```

Output

```
VA
CA
TI
ON
```

Explanation of output:
Pointer `p` points to character with even index, that is, `*p` will be character at even index. Pointer `p+1` points to character with odd index, that is, `*(p+1)` will be character at odd index. Every line displays two character, character at even index and character at odd index.

ii)
```
#include<iostream.h>
void main()
{
   int arr[]={23, 45, 67, 89, 34, 56, 78, 93};
   int *ptr=arr, p=1, s=0;
   for (int z=0; z<8; z++, ptr++)
   {
      int x=*ptr/10+*ptr%10;
      cout<<x<<endl;
      z%2==0 ? p*=*ptr/10 : s+=*ptr%10*10;
   }
   cout<<p<<","<<s<<endl;
}
```

Output

```
5
9
13
17
7
11
15
12
252,230
```

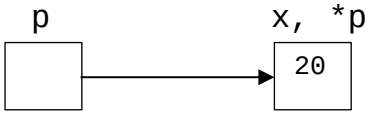
Explanation of output:
Variable **x** is the sum of the two digits (expression ***ptr/10** represent digit at tenth place and expression ***ptr%10** represent digit at unit place). Inside the function, sum of the two digits are displayed on the screen.
Accumulator **p** accumulate product of digits at the tenth place (***ptr/10**) if the number has even index.
Accumulator **s** accumulate sum of 10 times the digits at the unit place (***ptr%10*10**) if the number has odd index.

b) Rewrite the complete program by removing all the syntax errors (underline the corrections): [2]

```
#include<fstream.h>
void main()
{
   ofstream fout("EID.TXT");
   //Or, fstream fout("EID.TXT",ios::out);
   char *str="Kuwait had 9 days of Eid Holidays";
   //Or, char str[]="Kuwait had 9 days of Eid Holidays";
   //Or, char str[34]="Kuwait had 9 days of Eid Holidays";
   for (int x=0; str[x]; x++)
      fout.put(str[x]);   //Or, fout<<str[x];
   fout.close();
}
```

5. a) Define pointer. What is NULL pointer? Explain dereferencing with a suitable example. [2]
Pointer is a special type variable containing an address of another variable.
NULL pointer contains the address 0 (0x0000000).
Dereferencing is indirectly accessing a memory location using a pointer variable.

```
int x=20;
int *p=&x;
cout<<*p<<endl;        //Displays 20
```



b) What is a dynamic variable? Name the memory management operators and show their use with proper examples. [2]
Dynamic variable is a variable to which memory is allocated and de-allocated during the run-time.
Memory management operators are **new** (for allocating memory) and **delete** (for de-allocating memory).

```
int *p=new int(100);   //memory allocated during run-time
cout<<*p<<endl;        //displays 100
delete p;              //memory de-allocated during run-time
```