**Third Round: (Advanced Programming Round)**  
Here they asked us to create a “Railway reservation system” and gave us 4 modules. The modules were:  
    1. Booking  
    2. Availability checking  
    3. Cancellation  
    4. Prepare chart  
We were asked to create the modules for representing each data first and to continue with the implementation phase.

1.       Finding the maximum subsequent sum in the array containing positive and negative numbers.

2.       Finding the nth element from the last in a singly linked list.

3.       Finding the element occurring odd number of time in an array. No extra space and O(n) time complexity.

Eg: {1,1,3,4,4,4,4}

Output : 3

4.       Checking if the two strings are anagrams.

Eg: Input: cat,act

Output: yes

5.       Finding if one word is a rotation of another. The time complexity should be O(1). You can use one function named indexOf(s1,s2) that returns 1 if s2 is a substring of s2 else return 0.

Eg: bose and sobe

6.       Finding the kth smallest element in an array without sorting the elements in the array.

7.       A set of activities will be given along with their starting time and ending time. Write a code to find the maximum number of activities that can be performed without overlapping. Also find all possible activity sets that can be performed.

Eg:

Act1 st: 0 et: 4

Act2 st: 5 et: 6

Act3: st:0 et: 7

Act4: st:6 et: 7

Output: {1,2,4} and {3} – sets

3 – max no of activities.

8.       Write a program to print the numbers in words.(upto four digits)

Eg:

100 – hundred

1023 –One Thousand two hundred and three.

3.       Third round was advanced programming Round. It has 2 questions and time allotted was 3 hrs.

The questions given were of higher difficulty level.

1.       Write a program to scan all files in a folder and count the occurrence of each word in it. Also create inverted index based on the occurrence of the word as well as the document in which it occurs maximum.

2.       Implement a social network scenario and satisfy the following the requirements. Can use any data structure.

         List the friends of the logged in user

         List the total no of online users.

         List the friends who are online at any time.

         Log in and log out functions.

1) Display Anagrams

1. Input :{tar,rat,banana,art,nabana,baby}  
     
   Output :  
       anagrams :{tar,rat,art} {banana,nabana}  
       others      :{baby}  
     
   2) The array containing both positive and negative elments,  
      Display Maximum sum of consequent positive elements  
     
   Input : {-1,11,15,-10,30}  
     
   Output :   
   Max Sum  : 30  
   Elements : {30}  
     
   Input : {25,-11,28,15,-10,30,11}  
     
   Output :   
   Max Sum  : 43  
   Elements : {28,15}  
     
   using only one temporary 1D array to store output.    
     
     
   3) Display given String into Following Format  
     
   Input  : First Second Third Fourth  
     
   Output : Fourth Third Second First  
     
   Using Recursion  
     
   4) Count the indivitual chars and spaces, identifies   
      which count is maximum and remove that max counted   
      chars from the current String.  
     
   Input  : abc Abc aaC  
     
   Output :  
   Number of count 'A' : 1  
   Number of count 'C' : 1  
   Number of count 'a' : 3  
   Number of count 'b' : 2  
   Number of count 'c' : 2  
   Number of count ' ' : 2  
     
   After reomvin Character :  
       bc Abc C  
     
   Write a program to scan all files in a folder and count the occurrence of each word in it. Also create inverted index based on the occurrence of the word as well as the document in which it occurs maximum.
2. Implement a social network scenario and satisfy the following the requirements. Can use any data structure.

* List the friends of the logged in user
* List the total no of online users.
* List the friends who are online at any time.
* Log in and log out functions.
* [Zoho Interview | Set 3 COMPLEX CODING](http://markandayannotes.blogspot.com/2015/08/zoho-interview-set-3-complex-coding.html)
* 1) Design a Call taxi booking application  
  -There are n number of taxi’s. For simplicity, assume 4. But it should work for any number of taxi’s.  
  -The are 6 points(A,B,C,D,E,F)  
  -All the points are in a straight line, and each point is 15kms away from the adjacent points.  
  -It takes 60 mins to travel from one point to another  
  -Each taxi charges Rs.100 minimum for the first 5 kilometers and Rs.10 for the subsequent kilometers.  
  -For simplicity, time can be entered as absolute time. Eg: 9hrs, 15hrs etc.  
  -All taxi’s are initially stationed at A.  
  -When a customer books a Taxi, a free taxi at that point is allocated  
  -If no free taxi is available at that point, a free taxi at the nearest point is allocated.  
  -If two taxi’s are free at the same point, one with lower earning is allocated  
  -Note that the taxi only charges the customer from the pickup point to the drop point. Not the distance it travels from an adjacent point to pickup the customer.  
  -If no taxi is free at that time, booking is rejected  
  Design modules for
* 1) Call taxi booking
* Input 1:
* Customer ID: 1
* Pickup Point: A
* Drop Point: B
* Pickup Time: 9
* Output 1:
* Taxi can be allotted.
* Taxi-1 is allotted
* Input 2:
* Customer ID: 2
* Pickup Point: B
* Drop Point: D
* Pickup Time: 9
* Output 1:
* Taxi can be allotted.
* Taxi-2 is allotted
* (Note: Since Taxi-1 would have completed its journey when second booking is done, so Taxi-2 from nearest point A which is free is allocated)
* Input 3:
* Customer ID: 3
* Pickup Point: B
* Drop Point: C
* Pickup Time: 12
* Output 1:
* Taxi can be allotted.
* Taxi-1 is allotted

* 2) Display the Taxi details
* Taxi No: Total Earnings:
* BookingID CustomerID From To PickupTime DropTime Amount
* Output:
* Taxi-1 Total Earnings: Rs. 400
* 1 1 A B 9 10 200
* 3 3 B C 12 13 200
* Taxi-2 Total Earnings: Rs. 350
* 2 2 B D 9 11 350

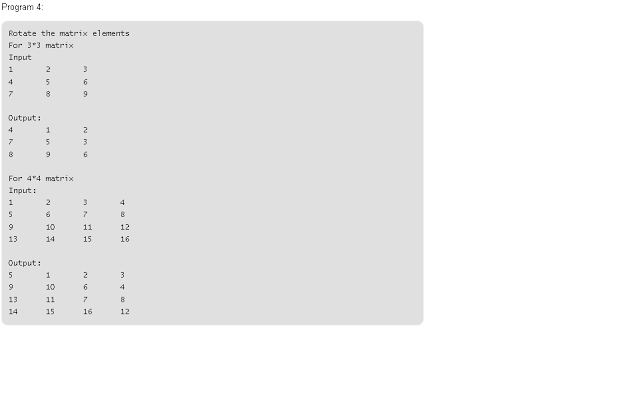
* These were just sample inputs. It should work for any input that they give.  
    
    
  **My Solution ::**
* #include <stdio.h>
* #include <string.h>
* #include <math.h>
* #include <stdlib.h>
* #include <conio.h>
* **struct** taxi
* {
* **int** c;
* **int** booking\_id[10];
* **int** customer\_id[10];
* **char** from[10];
* **char** to[10];
* **int** pickup\_time[10];
* **int** drop\_time[10];
* **int** amount[10];
* **char** current\_position;
* **int** current\_time;
* }s[4];
* **struct** customer
* {
* **int** customer\_id;
* **char** pickup\_point;
* **char** drop\_point;
* **int** pickup\_time;
* }cc;
* **int** cust()
* {
* **int** j;
* scanf("%d%s%s%d",&cc.customer\_id,&cc.pickup\_point,&cc.drop\_point,&cc.pickup\_time);
* j=calc();
* **return** j;
* }
* **int** tot(**int** i)
* {
* **int** k,total=0;
* **for**(k=0;k<s[i].c;k++)
* total=total+s[i].amount[k];
* **return** total;
* }
* **int** calc()
* {
* **int** k,min=100,i,m=0,j,total1,total2;
* **for**(i=0;i<4;i++)
* {
* k=abs(s[i].current\_position-cc.pickup\_point);
* **if**(min>k&&(s[i].current\_time<=cc.pickup\_time))
* {
* min=k;
* m=i;
* }
* **else** **if**(min==k&&(s[i].current\_time<=cc.pickup\_time))
* {
* total1=tot(m);
* total2=tot(i);
* **if**(total1>total2)
* {
* min=k;
* m=i;
* }
* }
* }
* **return** m;
* }
* **void** assign(**int** j)
* {
* s[j].booking\_id[s[j].c]=1;
* s[j].current\_position=cc.drop\_point;
* s[j].amount[s[j].c]=(((abs(cc.pickup\_point-cc.drop\_point)\*15)-5)\*10)+100;
* s[j].pickup\_time[s[j].c]=cc.pickup\_time;
* s[j].drop\_time[s[j].c]=(abs(cc.pickup\_point-cc.drop\_point)\*1)+cc.pickup\_time;
* s[j].from[s[j].c]=cc.pickup\_point;
* s[j].to[s[j].c]=cc.drop\_point;
* s[j].customer\_id[s[j].c]=cc.customer\_id;
* s[j].current\_time=s[j].drop\_time[s[j].c];
* s[j].c++;
* }
* **void** details()
* {
* **int** i,j,sum=0;
* scanf("%d",&j);
* j=j-1;
* **for**(i=0;i<s[j].c;i++)
* {
* sum=sum+s[j].amount[i];
* }
* printf("total:%d\n",sum);
* printf("\ntaxi:\tbooking id:\tcustomer id:\tfrom:\tto:\tpickup time:\tdrop time:\tamount:\tcurrent point:\n");
* **for**(i=0;i<s[j].c;i++)
* printf("%d\t%d\t%d\t%c\t%c\t%d\t%d\t%d\t%c",j,s[j].booking\_id[i],s[j].customer\_id[i],s[j].from[i],s[j].to[i],s[j].pickup\_time[i],s[j].drop\_time[i],s[j].amount[i],s[j].current\_position);
* }
* **int** **main**() {
* **int** i,j,k;
* **for**(i=0;i<4;i++)
* {
* s[i].c=0;
* s[i].current\_position='a';
* s[i].current\_time=0;
* }
* clrscr();
* **while**(1)
* {
* printf("\n1.customer\n2.display\n3.exit\n");
* scanf("%d",&k);
* **switch**(k)
* {
* **case** 1:
* j=cust();
* assign(j);
* **break**;
* **case** 2:
* details();
* **break**;
* **case** 3:
* exit(0);
* **default**:
* printf("invalid input");
* **break**;
* }
* }
* getch();
* **return** 0;
* }
* }

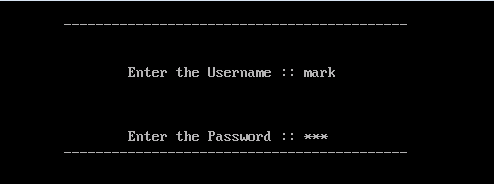
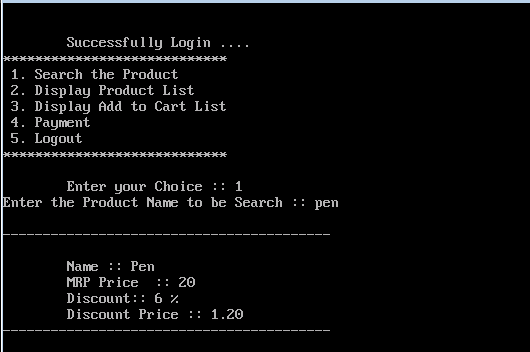
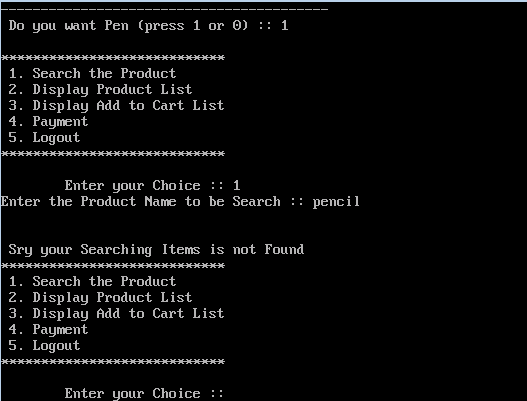
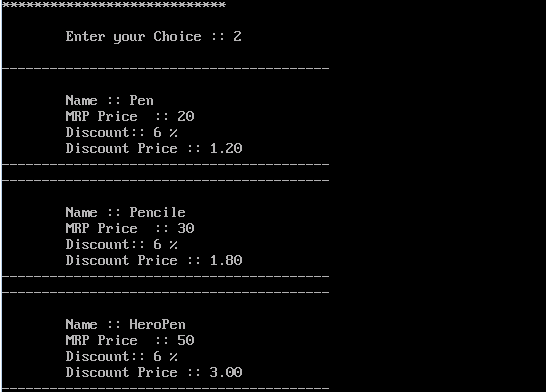
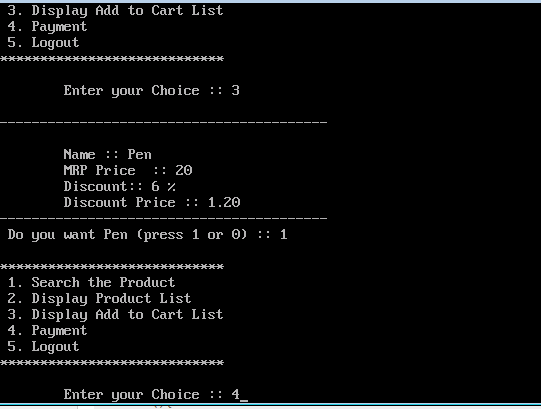
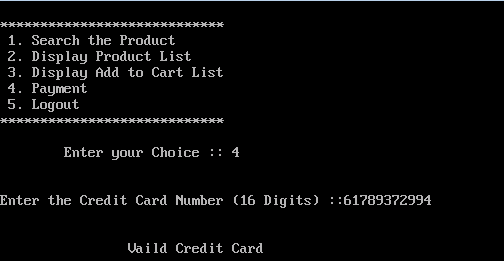
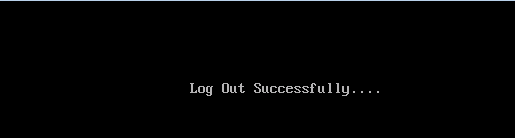
# [Zoho Interview | Set 1 (Advanced Programming Round)](http://markandayannotes.blogspot.com/2015/08/zoho-interview-set-1-advanced.html)

### Third Round: (Advanced Programming Round)

* Here they asked us to create a “Railway reservation system” and gave us 4 modules. The modules were:  
      1. Booking  
      2. Availability checking  
      3. Cancellation  
      4. Prepare chart  
  We were asked to create the modules for representing each data first and to continue with the implementation phase.  
    
  **My Solution :**
* #include<stdio.h>
* #include<conio.h>
* #include<stdlib.h>
* #include<string.h>
* #include<iostream.h>
* #include<time.h>
* #include<iomanip.h>
* #include<fstream.h>
* **char** f[10]="f";
* **char** s[10]="s";
* **int** addr,ad,flag,f1,d,m,i,amt;
* **float** tamt;
* class login
* {
* public:
* **char** id[100];
* **char** pass[100];
* **char** \*password;
* **void** getid()
* {
* cout<<"Enter your id:";
* gets(id);
* password=getpass("Enter the password:");
* strcpy(pass,password);
* }
* **void** displayid()
* {
* cout<<"Id:";
* puts(id);
* cout<<"Password:";
* puts(pass);
* }
* };
* class detail
* {
* public:
* **int** tno;
* **char** tname[100];
* **char** bp[100];
* **char** dest[100];
* **int** c1,c1fare;
* **int** c2,c2fare;
* **int** d,m,y;
* **void** getdetail()
* {
* cout<<"Enter the details as follows\n";
* cout<<"Train no:";
* cin>>tno;
* cout<<"Train name:";
* gets(tname);
* cout<<"Boarding point:";
* gets(bp);
* cout<<"Destination pt:";
* gets(dest);
* cout<<"No of seats in first class & fare per ticket:";
* cin>>c1>>c1fare;
* cout<<"No of seats in second class & fare per ticket:";
* cin>>c2>>c2fare;
* cout<<"Date of travel:";
* cin>>d>>m>>y;
* }
* **void** displaydetail()
* {
* cout<<tno<<"\t"<<tname<<"\t"<<bp<<"\t"<<dest<<"\t";
* cout<<c1<<"\t"<<c1fare<<"\t"<<c2<<"\t"<<c2fare<<"\t";
* cout<<d<<"-"<<m<<"-"<<y<<"\t"<<endl;
* }
* };
* class reser
* {
* public:
* **int** pnr;
* **int** tno;
* **char** tname[100];
* **char** bp[10];
* **char** dest[100];
* **char** pname[10][100];
* **int** age[20];
* **char** clas[10];
* **int** nosr;
* **int** i;
* **int** d,m,y;
* **int** con;
* **float** amc;
* **void** getresdet()
* {
* cout<<"Enter the details as follows\n";
* cout<<"Train no:";
* cin>>tno;
* cout<<"Train name:";
* gets(tname);
* cout<<"Boarding point:";
* gets(bp);
* cout<<"Destination pt:";
* gets(dest);
* cout<<"No of seats required:";
* cin>>nosr;
* **for**(i=0; i<nosr ; i++)
* {
* cout<<"Passenger name:";
* gets(pname[i]);
* cout<<"Passenger age:";
* cin>>age[i];
* }
* cout<<"Enter the class f-first class s-second class:";
* gets(clas);
* cout<<"Date of travel:";
* cin>>d>>m>>y;
* cout<<"Enter the concession category\n";
* cout<<"1.Military\n2.Senior citizen\n";
* cout<<"3.Children below 5 yrs\n4.None\n";
* cin>>con;
* cout<<"............END OF GETTING DETAILS............\n";
* }
* **void** displayresdet()
* {
* cout<<"...............................................\n";
* cout<<"...............................................\n";
* cout<<"Pnr no:"<<pnr;
* cout<<"\nTrain no:"<<tno;
* cout<<"\nTrain name:";
* puts(tname);
* cout<<"Boarding point:";
* puts(bp);
* cout<<"Destination pt:";
* puts(dest);
* cout<<"No of seats reserved:"<<nosr;
* **for**(i=0; i<nosr; i++)
* {
* cout<<"Passenger name:";
* puts(pname[i]);
* cout<<"Passenger age:"<<age[i];
* }
* cout<<"\nYour class:";
* puts(clas);
* cout<<"\nDate of reservation:"<<d<<"-"<<m<<"-"<<y;
* cout<<"\nYour concession category:"<<con;
* cout<<"\nYou must pay:"<<amc<<endl;
* cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";
* cout<<".........END OF RESERVATION.................\n";
* cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";
* }
* };
* class canc
* {
* public:
* **int** pnr;
* **int** tno;
* **char** tname[100];
* **char** bp[10];
* **char** dest[100];
* **char** pname[10][100];
* **int** age[20];
* **int** i;
* **char** clas[10];
* **int** nosc;
* **int** d,m,y;
* **float** amr;
* **void** getcancdet()
* {
* cout<<"Enter the details as follows\n";
* cout<<"Pnr no:";
* cin>>pnr;
* cout<<"Date of cancellation:";
* cin>>d>>m>>y;
* cout<<"...........END OF GETTING DETAILS...........\n";
* }
* **void** displaycancdet()
* {
* cout<<"...........................................\n";
* cout<<"...........................................\n";
* cout<<"Pnr no:"<<pnr;
* cout<<"\nTrain no:"<<tno;
* cout<<"\nTrain name:";
* puts(tname);
* cout<<"Boarding point:";
* puts(bp);
* cout<<"Destination pt:";
* puts(dest);
* cout<<"\nYour class:";
* puts(clas);
* cout<<"no of seats to be cancelled:"<<nosc;
* **for**(i=0; i<nosc; i++)
* {
* cout<<"Passenger name:";
* puts(pname[i]);
* cout<<"passenger age:"<<age[i];
* }
* cout<<"\nDate of cancellation:"<<d<<"-"<<m<<"-"<<y;
* cout<<"\nYou can collect:"<<amr<<"rs"<<endl;
* cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";
* cout<<".........END OF CANCELLATION.............\n";
* cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";
* }
* };
* **void** manage();
* **void** can();
* **void** user();
* **void** database();
* **void** res();
* **void** reserve();
* **void** displaypassdetail();
* **void** cancell();
* **void** enquiry();
* **void** **main**()
* {
* clrscr();
* **int** ch;
* cout<<"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
* cout<<".......WELCOME TO RAILWAY RESERVATION SYSTEM..........\n";
* cout<<"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
* **do**
* {
* cout<<"^^^^^^^^^^^^^^^^^^^^^^MAIN MENU^^^^^^^^^^^^^^^^^^^^\n";
* cout<<"1.Admin mode\n2.User mode\n3.Exit\n";
* cout<<"Enter your choice:";
* cin>>ch;
* cout<<endl;
* **switch**(ch)
* {
* **case** 1:
* database();
* **break**;
* **case** 2:
* user();
* **break**;
* **case** 3:
* exit(0);
* }
* }
* **while**(ch<=3);
* getch();
* }
* **void** database()
* {
* **char** \*password;
* **char** \*pass="codewithc";
* password=getpass("Enter the admininistrator password:");
* detail a;
* fstream f;
* **int** ch;
* **char** c;
* **if**(strcmp(pass,password)!=0)
* {
* cout<<"Enter the password correctly \n";
* cout<<"You are not permitted to logon this mode\n";
* goto h;
* }
* **if**(strcmp(pass,password)==0)
* {
* **char** c;
* **do**
* {
* cout<<"...........ADMINISTRATOR MENU...........\n";
* cout<<"1.Create detail data base\n2.Add details\n";
* cout<<"3.Display details\n4.User management\n";
* cout<<"5.Display passenger details\n6.Return to main menu\n";
* cout<<"Enter your choice:";
* cin>>ch;
* cout<<endl;
* **switch**(ch)
* {
* **case** 1:
* f.open("t.txt",ios::out|ios::binary);
* **do**
* {
* a.getdetail();
* f.write((**char** \*) & a,**sizeof**(a));
* cout<<"Do you want to add one more record?\n";
* cout<<"y-for Yes\nn-for No\n";
* cin>>c;
* }
* **while**(c=='y');
* f.close();
* **break**;
* **case** 2:
* f.open("t.txt",ios::in|ios::out|ios::binary|ios::app);
* a.getdetail();
* f.write((**char** \*) & a,**sizeof**(a));
* f.close();
* **break**;
* **case** 3:
* f.open("t.txt",ios::in|ios::out|ios::binary|ios::app);
* f.seekg(0);
* **while**(f.read((**char** \*) & a,**sizeof**(a)))
* {
* a.displaydetail();
* }
* f.close();
* **break**;
* **case** 4:
* manage();
* **break**;
* **case** 5:
* displaypassdetail();
* **break**;
* }
* }
* **while**(ch<=5);
* f.close();
* }
* h:
* }
* **void** reserve()
* {
* **int** ch;
* **do**
* {
* cout<<"1.Reserve\n2.Return to the main menu\n";
* cout<<"Enter your choice:";
* cin>>ch;
* cout<<endl;
* **switch**(ch)
* {
* **case** 1:
* res();
* **break**;
* }
* }
* **while**(ch==1);
* getch();
* }
* **void** res()
* {
* detail a;
* reser b;
* fstream f1,f2;
* time\_t t;
* f1.open("t.txt",ios::in|ios::out|ios::binary);
* f2.open("p.txt",ios::in|ios::out|ios::binary|ios::app);
* **int** ch;
* b.getresdet();
* **while**(f1.read((**char** \*) &a,**sizeof**(a)))
* {
* **if**(a.tno==b.tno)
* {
* **if**(strcmp(b.clas,f)==0)
* {
* **if**(a.c1>=b.nosr)
* {
* amt=a.c1fare;
* addr=f1.tellg();
* ad=**sizeof**(a.c1);
* f1.seekp(addr-(7\*ad));
* a.c1=a.c1-b.nosr;
* f1.write((**char** \*) & a.c1,**sizeof**(a.c1));
* **if**(b.con==1)
* {
* cout<<"Concession category:MILITARY PERSONNEL\n";
* b.amc=b.nosr\*((amt\*50)/100);
* }
* **else** **if**(b.con==2)
* {
* cout<<"Concession category:SENIOR CITIZEN\n";
* b.amc=b.nosr\*((amt\*60)/100);
* }
* **else** **if**(b.con==3)
* {
* cout<<"Concession category:CHILDERN BELOW FIVE\n";
* b.amc=0.0;
* }
* **else** **if**(b.con==4)
* {
* cout<<"You cannot get any concession\n";
* b.amc=b.nosr\*amt;
* }
* srand((**unsigned**) time(&t));
* b.pnr=rand();
* f2.write((**char** \*) & b,**sizeof**(b));
* b.displayresdet();
* cout<<"------------------------------------------------------\n";
* cout<<"--------------Your ticket is reserved-----------\n";
* cout<<"-----------------End of reservation menu-------\n";
* }
* **else**
* {
* cout<<"\*\*\*\*\*\*\*\*\*\*Sorry req seats not available\*\*\*\*\*\*\*\*\n";
* }
* }
* **else** **if**(strcmp(b.clas,s)==0)
* {
* **if**(a.c2>=b.nosr)
* {
* amt=a.c2fare;
* addr=f1.tellg();
* ad=**sizeof**(a.c2);
* f1.seekp(addr-(5\*ad));
* a.c2=a.c2-b.nosr;
* f1.write((**char** \*) & a.c2,**sizeof**(a.c2));
* **if**(b.con==1)
* {
* cout<<"Concession category:MILITARY PRESONNEL\n";
* b.amc=b.nosr\*((amt\*50)/100);
* }
* **else** **if**(b.con==2)
* {
* cout<<"Concession category:SENIOR CITIZEN\n";
* b.amc=b.nosr\*((amt\*60)/100);
* }
* **else** **if**(b.con==3)
* {
* cout<<"Concession category:CHILDERN BELOW FIVE\n";
* b.amc=0.0;
* }
* **else** **if**(b.con==4)
* {
* cout<<"You cannot get any concession\n";
* b.amc=b.nosr\*amt;
* }
* f2.write((**char** \*) & b,**sizeof**(b));
* b.displayresdet();
* cout<<"---------------------------------------\n";
* cout<<"--------Your ticket is reserved--------\n";
* cout<<"------------End of reservation---------\n";
* }
* **else**
* {
* cout<<"\*\*\*\*\*\*\*\*Sorry req no of seats not available\*\*\*\*\*\*\*\n";
* }
* }
* getch();
* goto h;
* }
* **else**
* {
* flag=0;
* }
* }
* **if**(flag==0)
* {
* cout<<"............Wrong train no......................\n";
* cout<<"......Enter the train no from the data base.....\n";
* }
* f1.close();
* f2.close();
* getch();
* h:
* }
* **void** displaypassdetail()
* {
* fstream f;
* reser b;
* f.open("p.txt",ios::in|ios::out|ios::binary);
* f.seekg(0);
* **while**(f.read((**char** \*) & b,**sizeof**(b)))
* {
* b.displayresdet();
* }
* f.close();
* getch();
* }
* **void** enquiry()
* {
* fstream f;
* f.open("t.txt",ios::in|ios::out|ios::binary);
* detail a;
* **while**(f.read((**char** \*) & a,**sizeof**(a)))
* {
* a.displaydetail();
* }
* getch();
* }
* **void** cancell()
* {
* detail a;
* reser b;
* canc c;
* fstream f1,f2,f3;
* f1.open("t.txt",ios::in|ios::out|ios::binary);
* f2.open("p.txt",ios::in|ios::out|ios::binary);
* f3.open("cn.txt",ios::in|ios::out|ios::binary);
* cout<<"\*\*\*\*\*\*\*\*\*\*CANCELLATION MENU\*\*\*\*\*\*\*\*\*\n";
* c.getcancdet();
* **while**(f2.read((**char** \*) & b,**sizeof**(b)))
* {
* **if**(b.pnr==c.pnr)
* {
* c.tno=b.tno;
* strcpy(c.tname,b.tname);
* strcpy(c.bp,b.bp);
* strcpy(c.dest,b.dest);
* c.nosc=b.nosr;
* **for**(**int** j=0; j<c.nosc; j++)
* {
* strcpy(c.pname[j],b.pname[j]);
* c.age[j]=b.age[j];
* }
* strcpy(c.clas,b.clas);
* **if**(strcmp(c.clas,f)==0)
* {
* **while**(f1.read((**char** \*) & a,**sizeof**(a)))
* {
* **if**(a.tno==c.tno)
* {
* a.c1=a.c1+c.nosc;
* d=a.d;
* m=a.m;
* addr=f1.tellg();
* ad=**sizeof**(a.c1);
* f1.seekp(addr-(7\*ad));
* f1.write((**char** \*) & a.c1,**sizeof**(a.c1));
* tamt=b.amc;
* **if**((c.d==d)&&(c.m==m))
* {
* cout<<"You are cancelling at the date of departure\n";
* c.amr=tamt-((tamt\*60)/100);
* }
* **else** **if**(c.m==m)
* {
* cout<<"You are cancelling at the month of departure\n";
* c.amr=tamt-((tamt\*50)/100);
* }
* **else** **if**(m>c.m)
* {
* cout<<"You are cancelling one month before the date of departure\n";
* c.amr=tamt-((tamt\*20)/100);
* }
* **else**
* {
* cout<<"Cancelling after the departure\n";
* cout<<"Your request cannot be completed\n";
* }
* goto h;
* c.displaycancdet();
* }
* }
* }
* **else** **if**(strcmp(c.clas,s)==0)
* {
* **while**(f1.read((**char** \*) & a,**sizeof**(a)))
* {
* **if**(a.tno==c.tno)
* {
* a.c2=a.c2+c.nosc;
* d=a.d;
* m=a.m;
* addr=f1.tellg();
* ad=**sizeof**(a.c2);
* f1.seekp(addr-(5\*ad));
* f1.write((**char** \*) & a.c2,**sizeof**(a.c2));
* tamt=b.amc;
* **if**((c.d==d)&&(c.m==m))
* {
* cout<<"You are cancelling at the date of departure\n";
* c.amr=tamt-((tamt\*60)/100);
* }
* **else** **if**(c.m==m)
* {
* cout<<"You are cancelling at the month of departure\n";
* c.amr=tamt-((tamt\*50)/100);
* }
* **else** **if**(m>c.m)
* {
* cout<<"You are cancelling one month before the date of departure\n";
* c.amr=tamt-((tamt\*20)/100);
* }
* **else**
* {
* cout<<"Cancelling after the departure\n";
* cout<<"Your request cannot be completed\n";
* }
* goto h;
* c.displaycancdet();
* }
* }
* }
* }
* **else**
* {
* flag=0;
* }
* }
* h:
* **if**(flag==0)
* {
* cout<<"Enter the correct pnr no\n";
* }
* f1.close();
* f2.close();
* f3.close();
* getch();
* }
* **void** can()
* {
* **int** ch;
* **do**
* {
* cout<<".................CANCELLATION MENU.........\n";
* cout<<"1.Cancell\n2.Return to the main menu\n";
* cout<<"Enter your choice:";
* cin>>ch;
* cout<<endl;
* **switch**(ch)
* {
* **case** 1:
* cancell();
* **break**;
* }
* }
* **while**(ch==1);
* getch();
* }
* **void** user()
* {
* login a;
* **int** ch;
* cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";
* cout<<"\*\*\*\*\*\*\*\*\*\*\*WELCOME TO THE USER MENU\*\*\n";
* cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";
* **char** \*password;
* fstream f;
* f.open("id.txt",ios::in|ios::out|ios::binary);
* **char** id[100];
* puts("Enter your id:");
* gets(id);
* password=getpass("Enter your password:");
* **while**(f.read((**char** \*) & a,**sizeof**(a)))
* {
* **if**((strcmp(a.id,id)==0)&&(strcmp(a.pass,password)==0))
* {
* **do**
* {
* cout<<"1.Reserve\n2.Cancell\n3.Enquiry\n4.Return to the main menu\n";
* cout<<"Enter your choice:";
* cin>>ch;
* cout<<endl;
* **switch**(ch)
* {
* **case** 1:
* reserve();
* **break**;
* **case** 2:
* cancell();
* **break**;
* **case** 3:
* enquiry();
* **break**;
* }
* }
* **while**(ch<=3);
* goto j;
* }
* **else**
* {
* d=1;
* }
* }
* **if**(d==1)
* {
* cout<<"Enter your user id and password correctly\n";
* }
* getch();
* j:
* }
* **void** manage()
* {
* **int** ch;
* fstream f;
* **char** c;
* login a;
* cout<<".........WELCOME TO THE USER MANAGEMENT MENU........\n";
* **do**
* {
* cout<<"1.Create id data base\n2.Add details\n";
* cout<<"3.Display details\n4.Return to the main menu\n";
* cout<<"Enter your choice:";
* cin>>ch;
* cout<<endl;
* **switch**(ch)
* {
* **case** 1:
* f.open("id.txt",ios::out|ios::binary);
* **do**
* {
* a.getid();
* f.write((**char** \*) & a,**sizeof**(a));
* cout<<"Do you want to add one more record\n";
* cout<<"y-Yes\nn-No\n";
* cin>>c;
* }
* **while**(c=='y');
* f.close();
* **break**;
* **case** 2:
* f.open("id.txt",ios::in|ios::out|ios::binary|ios::app);
* a.getid();
* f.write((**char** \*) & a,**sizeof**(a));
* f.close();
* **break**;
* **case** 3:
* f.open("id.txt",ios::in|ios::out|ios::binary);
* f.seekg(0);
* **while**(f.read((**char** \*) & a,**sizeof**(a)))
* {
* a.displayid();
* }
* f.close();
* **break**;
* }
* }
* **while**(ch<=3);
* getch();
* }

#### 

* [ZOHO find new friends in social network ??](http://markandayannotes.blogspot.com/2015/07/zoho-find-new-friends-in-social-network.html)
* **Help john to find new friends in social network**  
    
  **Input:**  
    
  3  
  Mani 3 ram raj guna  
  Ram 2 kumar Kishore  
  Mughil 3 praveen Naveen Ramesh   
    
    
  **Output:**  
    
  Raj guna kumar Kishore praveen Naveen Ramesh
* #include<stdio.h>
* #include<conio.h>
* #include<string.h>
* #define N 10
* **struct** Node{
* **char** name[20];
* **char** list[N][20];
* **int** no;
* }friends[N];
* **void** findFriend(**int** n){
* **int** i,j,k,count,m;
* **for**(i=0;i<n;i++){
* m = friends[i].no;
* **for**(j=0;j<m;j++){
* **for**(k=count=0;k<n;k++){
* **if**( strcmpi(friends[i].list[j],friends[k].name) != 0){
* count++;
* }
* }
* **if**(count==n)
* printf("%s ",friends[i].list[j]);
* }
* }
* }
* **int** **main**(){
* **int** t,n,i,j;
* clrscr();
* scanf("%d",&t);
* **for**(i=0;i<t;i++){
* scanf(" %s",friends[i].name);
* scanf(" %d",&n);
* friends[i].no = n;
* **for**(j=0;j<n;j++){
* scanf(" %s",friends[i].list[j]);
* }
* }
* findFriend(t);
* getch();
* }
* [Zoho Interview | Set 5 (On-Campus Drive)](http://markandayannotes.blogspot.com/2015/07/zoho-interview-set-5-on-campus-drive.html)
* [](http://3.bp.blogspot.com/-NQ33CkNqxlg/VbJiW03uchI/AAAAAAAAAFs/sUCMfBreQyg/s1600/untitled.PNG)


* #include<stdio.h>
* #include<conio.h>
* #define N 6
* **void** display(**int** a[N][N]){
* **int** i,j;
* **for**(i=0;i<N;i++){
* **for**(j=0;j<N;j++)
* printf("%4d",a[i][j]);
* printf("\n\n");
* }
* }
* **void** rotateMatrix(**int** a[N][N], **int** b[N][N]){
* **int** i,j,temp,times,end,start;
* times = 0;
* end = N-1;
* start = 0;
* temp = N/2;
* **while**(times < temp){
* i = j = times;
* b[i][j++] = a[i+1][j];
* **while**( j < end ){
* b[i][j++] = a[i][j-1];
* }
* b[i++][j] = a[i][j-1];
* **while**( i < end ){
* b[i++][j] = a[i-1][j];
* }
* b[i][j--] = a[i-1][j];
* **while**( j > start ){
* b[i][j--] = a[i][j+1];
* }
* b[i--][j] = a[i][j+1];
* **while**( i > start ){
* b[i--][j] = a[i+1][j];
* }
* times++;
* end--;
* start++;
* }
* }
* **int** main(){
* **int** a[N][N],i,j,b[N][N],k;
* clrscr();
* memset(b, 0, **sizeof**(b));
* k = 1;
* **for**(i=0;i<N;i++)
* **for**(j=0;j<N;j++)
* a[i][j] = k++;
* display(a);
* rotateMatrix(a, b);
* printf("\n\nAfter the Rotate...\n\n\n");
* display(b);
* getch();
* **return** 0;
* }
* [e-Shop](http://markandayannotes.blogspot.com/2015/07/e-shop.html)
* #include<stdio.h>
* #include<conio.h>
* #include<stdlib.h>
* #include<string.h>
* #define N 3
* **struct** Product{
* **char** name[20];
* **int** price;
* **int** discount;
* **int** selected;
* }product[N];
* **struct** Customer{
* **char** name[20];
* **struct** Product list[N];
* **char** credit[20];
* **double** amount;
* }cust;
* **void** intialProductList();
* **void** displayList();
* **int** getDiscount(){
* srand(time(NULL));
* **return** (rand()%6+1);
* }
* **double** discountPrice(**struct** Product node){
* **double** temp;
* //printf("\n>>%f",k);
* temp = ((**double**)node.price/(**double**)100) \* node.discount ;
* //printf("---%lf",temp);
* **return** temp;
* }
* **void** display(**struct** Product product){
* printf("\n-----------------------------------------");
* printf("\n\n\tName :: %s",product.name);
* printf("\n\tMRP Price :: %d",product.price);
* printf("\n\tDiscount:: %d %%",product.discount);
* printf("\n\tDiscount Price :: %.2lf",discountPrice(product));
* printf("\n-----------------------------------------");
* getch();
* }
* **void** displayList(){
* **int** i;
* **for**(i=0;i<N;i++)
* display(product[i]);
* }
* **void** displayCart(**char** \*name){
* **int** i,k;
* k = 0;
* strcpy(cust.name,name);
* cust.amount = 0;
* **for**(i=0;i<N;i++)
* **if**( product[i].selected == 1){
* display(product[i]);
* printf("\n Do you want %s (press 1 or 0) :: ",product[i].name);
* scanf("%d",&product[i].selected);
* **if**(product[i].selected){
* cust.list[k++] = product[i];
* cust.amount += discountPrice(product[i]);
* }
* }
* }
* **void** initalProductList(){
* FILE \*fptr;
* **int** i;
* clrscr();
* **if**( (fptr=fopen("tempfile.txt","r")) == NULL )
* printf("OOPSS ......File cannot be open ");
* **else**{
* **for**(i=0;i<N;i++){
* fscanf(fptr,"%s",product[i].name);
* fscanf(fptr,"%d",&product[i].price);
* product[i].discount = getDiscount();
* product[i].selected = 0;
* }
* }
* fclose(fptr);
* }
* **int** isValid(**char** \*name, **char** \*password){
* FILE \*fptr;
* **char** n[20],p[20];
* clrscr();
* **if**( (fptr=fopen("log.txt","r")) == NULL )
* printf("\n\n\tOOPSS ......Login File Missing");
* **else**{
* **while**(!feof(fptr)){
* fscanf(fptr,"%s",n);
* fscanf(fptr,"%s",p);
* **if**( (strcmp(n,name)==0) && (strcmp(p,password)==0) ){
* fclose(fptr);
* **return** 1;
* }
* }
* }
* fclose(fptr);
* **return** 0;
* }
* **int** isValidProduct(**char** \*temp){
* **int** i;
* **for**(i=0;i<N;i++){
* **if**( (strcmpi(product[i].name, temp)) == 0 )
* **return** i;
* }
* **return** -1;
* }
* **int** sumDigits(**int** n){
* //printf("< %d > ",n%9);
* **if**((n%9)==0)
* **return** 9;
* **return** (n%9);
* }
* **int** chartonum(**char** ch){
* **return** (ch-'0');
* }
* **int** isValidCredit(**char** \*temp){
* **int** i,sum;
* strrev(temp);
* //puts(temp);
* **for**(sum = i =0; i<strlen(temp); i++){
* **if**(i%2==1)
* sum += sumDigits(chartonum(temp[i]) \* 2);
* **else**
* sum += chartonum(temp[i]);
* //printf("%3d",sum);
* }
* **if**( (sum%10) == 0 )
* **return** 1;
* **return** 0;
* }
* **void** userInterface(**char** \*name){
* **int** ch,temp[20];
* **while**(1){
* printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");
* printf("\n 1. Search the Product");
* printf("\n 2. Display Product List");
* printf("\n 3. Display Add to Cart List");
* printf("\n 4. Payment");
* printf("\n 5. Logout");
* printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");
* printf("\n\n\tEnter your Choice :: ");
* scanf("%d",&ch);
* **switch**(ch){
* **case** 1:
* printf("Enter the Product Name to be Search :: ");
* scanf("%s",temp);
* ch = isValidProduct(temp);
* **if**(ch > -1){
* display(product[ch]);
* printf("\n Do you want %s (press 1 or 0) :: ",product[ch].name);
* scanf("%d",&product[ch].selected);
* }
* **else**
* printf("\n\n Sry your Searching Items is not Found");
* **break**;
* **case** 2:
* displayList();
* **break**;
* **case** 3:
* displayCart(name);
* **break**;
* **case** 4:
* printf("\n\nEnter the Credit Card Number (16 Digits) :: ");
* scanf("%s",temp);
* **if**(isValidCredit(temp))
* printf("\n\n\t\tVaild Credit Card ");
* **else**
* printf("\n\n\t\tNOT Vaild Credit Card ");
* **break**;
* **case** 5:
* clrscr();
* printf("\n\n\n\n\n\t\t\tLog Out Successfully....");
* getch();
* exit(0);
* }
* }
* }
* **void** getLogin(){
* **char** name[20],pwd[20],ch,i=0;
* printf("\n\t-------------------------------------------");
* printf("\n\n\n\t\tEnter the Username :: ");
* scanf("%[^'\n']s",name);
* printf("\n\n\n\t\tEnter the Password :: ");
* **while**( (ch=getch()) != 13 ){
* printf("\*");
* pwd[i++] = ch;
* }
* pwd[i] = '\0';
* printf("\n\t-------------------------------------------\n");
* // puts(name);puts(pwd);
* getch();
* **if**( isValid(name,pwd) ){
* printf("\n\n\tSuccessfully Login ....");
* //displayList();
* userInterface(name);
* }
* **else**{
* clrscr();
* printf("\n\n\tInvailed Username or Password");
* getch();
* getLogin();
* }
* }
* **int** **main**(){
* clrscr();
* initalProductList();
* getLogin();
* getch();
* **return** 0;
* }
* [](http://4.bp.blogspot.com/-t9IXzryuEA8/VbHiM_lzbzI/AAAAAAAAAEo/4Sd_lOtvgA4/s1600/1.PNG)
* [](http://2.bp.blogspot.com/-jmBxnkiQtAY/VbHiMwXXQKI/AAAAAAAAAEw/Dqxsot-EpnM/s1600/2.PNG)
* [](http://2.bp.blogspot.com/-hLSYIs28T_Y/VbHiM_lELuI/AAAAAAAAAEs/JqI_h3Hhk7s/s1600/3.PNG)
* [](http://1.bp.blogspot.com/-jR9jU8Qi3j0/VbHiNoY8_9I/AAAAAAAAAE0/wxULGNJINk4/s1600/4.PNG)
* [](http://3.bp.blogspot.com/-7wbE8LA-4iY/VbHiNn8JxyI/AAAAAAAAAE4/NoIIgsAsNKs/s1600/5.PNG)
* [](http://2.bp.blogspot.com/-dNtrzYtkNKU/VbHiN5xcQNI/AAAAAAAAAE8/m2x9TYezu4o/s1600/6.PNG)
* [](http://4.bp.blogspot.com/-pinaRy8eisY/VbHiONMdonI/AAAAAAAAAFA/qsaClx1Uwxg/s1600/7.PNG)

# [Zoho Interview Round three ( inventory stock list )](http://markandayannotes.blogspot.com/2015/07/zoho-interview-round-three-inventory.html)

* **1) To form a structure which has few elements:**
* **struct product {**
* **char productname[20];**
* **int product\_price;**
* **int product\_id;**
* **}**
* **Get the product name, price and id and display the product name and price in descending of the price.**
* **2) For the same above structure, now add another structure which is the category. That category will have products in it.**
* **Struct category**
* **{**
* **char category\_name[20];**
* **int cat\_id;**
* **}**
* **According the category get the product name, product price and id, then display all the products category wise in descending order.**
* **3) For the same structure which as category and product, get the category id from the user in the product structure and save to the category list. Then display them all in category wise.**  
   **4) A sheet full of data will be given with inventory stock list, which as different categories and different products as input with category capacity and product availability in the structure. Now we need to add a new category or new product with capacity and availability. Need to check whether the product availability is exceeding the category capacity, if yes the output rack is full or else tell how much free space is available and add the product to lis**t.
* **My Solution :::**
* #include<stdio.h>
* #include<conio.h>
* #include<string.h>
* #define N 4
* #define M 3
* **int** availableCategory = 0;
* **struct** Product{
* **char** product\_name[20];
* **int** product\_price;
* **int** product\_id;
* **int** isallocated;
* };
* **struct** Category{
* **char** category\_name[20];
* **char** category\_id;
* **struct** Product product[N];
* **int** isallocated;
* **int** number\_of\_product;
* };
* **void** insertCategory(**struct** Category \*);
* **void** insertProduct(**struct** Product \*);
* **void** swapProduct(**struct** Product \*, **struct** Product \*);
* **void** swapCategory(**struct** Category \*, **struct** Category \*);
* **void** sortProduct(**struct** Product [N]);
* **void** sortCategory(**struct** Category [M]);
* **void** displayAllProduct(**struct** Product [N]);
* **void** displayProduct(**struct** Product);
* **void** displayCategory(**struct** Category);
* **void** displayAllCategory(**struct** Category [M]);
* **int** find(**struct** Category \*, **int**);
* **void** search();
* **void** search(**struct** Category \*root){
* **int** id;
* printf("\n\n Enter the Category ID:");
* scanf("%d",&id);
* **if**(find(root,id) != -1)
* displayCategory(root[find(root,id)]);
* **else**
* printf("\n\n\t\tOops ...! Not Found");
* }
* **void** insertProduct(**struct** Product \*newnode){
* printf("\n\n\t\tEnter the Product Name :: ");
* scanf(" %s",newnode->product\_name);
* printf("\n\n\t\tEnter the Product ID :: ");
* scanf(" %d",&newnode->product\_id);
* printf("\n\n\t\tEnter the Product Price :: ");
* scanf(" %d",&newnode->product\_price);
* }
* **void** insertCategory(**struct** Category \*newnode){
* **int** id,i;
* printf("\n\n Enter the Category ID:");
* scanf("%d",&id);
* **if**( ( i = find(newnode,id) ) != -1){
* **if**(newnode[i].number\_of\_product > N){
* printf("\n\n\t\tNo More Capacity for Category");
* **return**;
* }
* insertProduct(&newnode[i].product[newnode[i].number\_of\_product]);
* newnode[i].number\_of\_product++;
* }
* **else**{
* **if**(availableCategory > M){
* printf("\n\n\t\tNo More Capacity for Category");
* **return**;
* }
* i = availableCategory;
* newnode[i].category\_id=id;
* printf("\n\n Enter the Category Name :: ");
* scanf("%s",newnode[i].category\_name);
* insertProduct(&newnode[i].product[newnode[i].number\_of\_product]);
* newnode[i].number\_of\_product++;
* availableCategory++;
* }
* }
* **int** find(**struct** Category \*node,**int** id){
* **int** i;
* **for**(i=0;i<M;i++)
* **if**(node[i].category\_id == id)
* **return** i;
* **return** -1;
* }
* **void** swapProduct(**struct** Product \*a, **struct** Product \*b){
* **struct** Product \*temp;
* memcpy(temp, a, **sizeof**(**struct** Product));
* memcpy(a, b, **sizeof**(**struct** Product));
* memcpy(b, temp, **sizeof**(**struct** Product));
* }
* **void** swapCategory(**struct** Category \*a, **struct** Category \*b){
* **struct** Category \*temp;
* memcpy(temp, a, **sizeof**(**struct** Category));
* memcpy(a, b, **sizeof**(**struct** Category));
* memcpy(b, temp, **sizeof**(**struct** Category));
* }
* **void** sortProduct(**struct** Product node[N]){
* **int** i,j;
* **for**(j=0;j<N;j++)
* **for**(i=j;i<N;i++)
* **if**(node[i].product\_id && node[j].product\_id)
* **if**(node[j].product\_price < node[i].product\_price)
* swapProduct(&node[j],&node[i]);
* }
* **void** sortCategory(**struct** Category node[M]){
* **int** i,j;
* **for**(i=0;i<N;i++)
* sortProduct(node[i].product);
* **for**(j=0;j<M;j++)
* **for**(i=j;i<M;i++)
* **if**(strcmp(node[j].category\_name, node[i].category\_name) > 0)
* swapCategory(&node[j],&node[i]);
* }
* **void** displayAllProduct(**struct** Product node[N]){
* **int** i;
* **for**(i=0;i<N;i++)
* **if**(node[i].product\_id)
* displayProduct(node[i]);
* }
* **void** displayProduct(**struct** Product node){
* printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");
* printf("\n\nProduct Name :: %s",node.product\_name);
* printf("\nProduct Price :: %d",node.product\_price);
* printf("\nProduct ID :: %d",node.product\_id);
* printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");
* getch();
* }
* **void** displayCategory(**struct** Category node){
* printf("\n$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$");
* printf("\n\nCategory Name :: %s",node.category\_name);
* printf("\nCategory ID :: %d",node.category\_id);
* displayAllProduct(node.product);
* printf("\n\n$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$");
* getch();
* }
* **void** displayAllCategory(**struct** Category node[M]){
* **int** i;
* **for**(i=0;i<M;i++)
* displayCategory(node[i]);
* }
* **int** **main**(){
* **int** ch;
* **struct** Category root[M];
* clrscr();
* **do**{
* printf("\n\n 1.Insert Product\n\n 2.Search Category ID\n\n 3.Display All Category\n\n 4.Exit (Enter -ve number)");
* printf("\n\n\n\t\tEnter your Choice :: ");
* scanf("%d",&ch);
* **switch**(ch){
* **case** 1:
* insertCategory(root);
* sortCategory(root);
* **break**;
* **case** 2:
* search(root);
* **break**;
* **case** 3:
* displayAllCategory(root);
* **break**;
* **case** 4:
* **return** 0;
* **default**:
* printf("Ooops ... Entered choice is wrong ...!");
* getch();
* **return** 0;
* }
* }**while**(1);
* getch();
* **return** 0;
* }
* [CounterCode 2015 ( Imba )](http://markandayannotes.blogspot.com/2015/08/countercode-2015-imba.html)
* **Problem Statement**
* A DOTA game has *N* heroes, each with a distinct rank from [1..*N*]. In DOTA every formation is characterized as a permutation [1...*N*] of ranks of players.   
  A formation is **Imba** when the sum of ranks of every two consecutive players is less than or equal to (*N*+1). Given *N*, you are to print the lexicographically smallest permutation of ranks [1...*N*] that makes the formation **Imba**.
* **Input Format**
* The first line will contain an integer *T*, i.e. the number of the test cases followed by *T* lines, each containing the value of *N*.  
    
  **Constraints**   
  1≤*T*≤5  
  2≤*N*≤105
* **Output Format**
* *T* lines each containing the permutation; the numbers in each line should be seperated by a single space.
* **Sample Input**
* 2
* 2
* 3
* **Sample Output**
* 1 2
* 2 1 3
* **Explanation**
* In the first case there are two possible permutations [1,2] and [2,1]. Both of the given permutations satisfy the given constraints and [1,2] is lexicographically smaller than [2,1].   
  In the second case, the two possible permutations are [2,1,3] and [3,1,2], of which the former is lexicographically smaller.
* #include <stdio.h>
* #include <string.h>
* #include <math.h>
* #include <stdlib.h>
* **int** **main**() {
* **int** t,n,temp,i;
* scanf("%d",&t);
* **while**(t--){
* scanf("%d",&n);
* **if**(n%2==0){
* temp = n/2;
* printf("%d ",temp);
* i = 1;
* **while**((temp+i)!=n){
* printf("%d %d ",(temp+i),(temp-i));
* i++;
* }
* }
* **else**{
* temp = (n/2)+1;
* i = 1;
* printf("%d ",temp);
* **while**((temp+i)!=n){
* printf("%d %d ",(temp-i),(temp+i));
* i++;
* }
* printf("%d ",temp-i);
* }
* printf("%d",n);
* printf("\n");
* }
* **return** 0;
* }

5. Consider a train ticket system. Maintain the passenger details for each  
train. Display train wise, berth wise, PNR number wise. Note: one PNR can  
contain many passengers.  
  
use only java..  
  
  
  
**ZOHO ROUND 3 – ADVANCED CODING - 3 HOURS**

Build a banking system to store all details about all customers and I was asked to do twelve tasks such as adding new customers, deleting customers, modify customer details, changing password, money withdrawal, money deposit, querying for customer details, list the customer having balance less than 1000.

Basic structure of customer details :

CustId AccountNo Name Balance EncryptedPwd

11 11011 Madhu 10000 AsthYujn

22 22022 Robin 20000 3Dga4mJ

33 33033 Raven 30000 2$gDhbc

Imagine this scenario: I/O completion ports are communications ports which take handles to files, sockets, or any other I/O. When a Read or Write is submitted to them, they cache the data (if necessary), and attempt to take the request to completion. Upon error or completion, they call a user-supplied function to let the users application know that that particular request has completed. They work asynchronously, and can process an unlimited number of simultaneous requests. Design the implementation and thread models for I/O completion ports. Remember to take into account multi-processor machines.

Suppose you have an array of 1001 integers. The integers are in random order, but you know each of the integers is between 1 and 1000 (inclusive). In addition, each number appears only once in the array, except for one number, which occurs twice. Assume that you can access each element of the array only once. Describe an algorithm to find the repeated number. If you used auxiliary storage in your algorithm, can you find an algorithm that does not require it

Beautiful You Pvt. Ltd. owns several parlors. Research has shown that if a customer arrives and there is no staff available to service them, the customer will turn around and leave, thus costing the company a sale. Your task is to write a program that tells the company how many customers left without getting any service.

Write a Java program to check whether the message sent has any error using the error detection method Cyclic Redundancy Code ( CRC ).

In this problem, each letter of the alphabet corresponds to a number using the scheme: a=1, b=2, c=3, ... y=25, z=26. To encode a message, an encryption key word is added to the message. The key word is the first word in the message that is five or more characters long. For example, if the message were:

"give me liberty or give me death," the key word would be "liberty."

The encrypted message would be: s r x j e y k u k g w l s n d p k a w g d p n c y z

give me liberty or give me death  
libe rt ylibert yl iber ty liber  
-------------------------------------------------------------------------------------  
srxj ey kukgwls nd pkaw gd pncyz