Q. 25. Write a program for the following problem:

INPUT: Positive integers $r_1; r_2; : : : ; r_n$ and $c_1; : : : ; c_n$.
OUTPUT: An n by n matrix A with 0/1 entries such that for all i the sum of the ith row in A is r_i and the sum of the ith column in A is c_i, if such a matrix exists.
Think of the problem this way. You want to put pawns on an n by n chessboard so that the ith row has r_i pawns and the ith column has c_i pawns.

Code:

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int a[100][100];
    int r[100],c[100],i,j;
    int n;
    cin>>n;
    for(i=0;i<n;i++)
        cin>>r[i];

    for(i=0;i<n;i++)
        cin>>c[i];
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            if(r[i]>0 && c[j]>0)
            {
                a[i][j]=1;
                r[i]--;
                c[j]--;
            }
        }
    }
    int k=0;
    for(i=0;i<n;i++)
    {
        if(c[i]!=0 || r[i]!=0)
        {
            k=1;
            break;
        }
    }

    if(k==0)
    {
        cout<<"Possible"<<endl;
        for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
                cout<<a[i][j]<<" ";
            cout<<endl;
        }
    }
    else
        cout<<"Not Possible";
    return 0;
}
```

This solution is correct due to the greedy choice that has been made in its algorithm.
At every step, we choose what is the best out of the given options and cumulatively, we obtain an optimal solution.

Q. 27. You wish to drive from point A to point B along a highway minimizing the time that you are stopped for gas. You are told beforehand the capacity C of you gas tank in liters, your rate F of fuel consumption in liters/kilometer, the rate r in liters/minute at which you can fill your tank at a gas station, and the locations A=x1,x2, ... , B=xn of the gas stations along the highway. So if you stop to fill your tank from 2 liters to 8 liters, you would have to stop for 6/r minutes. Consider the following two algorithms:

(a) Stop at every gas station, and fill the tank with just enough gas to make it to the next gas station.
(b) Stop if and only if you don't have enough gas to make it to the next gas station, and if you stop, fill the tank up all the way.

For each algorithm write a program and also either prove or disprove that this algorithm correctly solves the problem. Your proof of correctness must use an exchange argument.

CODE:

```
//Part 1
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int c,r,n,a[1000],0;
    double ans=0.0,f;
    cin>>c>>f>>r;
    cin>>n;
    for(i=0;i<n;i++)
        cin>>a[i];
    for(i=0;i<n-1;i++)
    {
        double x=a[i+1]-a[i];
        ans=ans+(x*f)/r;
    }
    cout<<ans<<endl;
}
```

CODE:

```
//Part 1
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int c,r,n,a[1000],i;
    double ans=0.0,f;
    cin>>c>>f>>r;
    cin>>n;
    for(i=0;i<n;i++)
        cin>>a[i];
    ans=ans+c/r;
    int p=c;
    for(i=1;i<n-1;i++)
    {
        int x=a[i+1]-a[i];
        p=p-x*f;
        if(p<0)
        {
            ans=ans+c/r;
            p=c;
        }
    }
```

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

Based on the results from above codes, it turns out that 2nd algorithm works better. This is because it uses Greedy approach. It makes the best decisions available at a time.

The problem with first part, i.e. stopping at every station and just filling enough gas for next station is that we have to stop at every station. But with fuel tanks with large capacity, this approach would prove to be disadvantageous.

Q. 29. Write a program for the following problem:

Consider the following bridge crossing problem where n people with speeds s1; : : : ; sn wish to cross the bridge as quickly as possible.
The rules remain:

- It is nighttime and you only have one ash-light.
- A maximum of two people can cross at any one time
- Any party who crosses, either 1 or 2 people must have the ash-light with them.
- The ash-light must be walked back and forth, it cannot be thrown, etc.
- A pair must walk together at the rate of the slower person's pace.

Give an efficient algorithm to find the fastest way to get a group of people across the bridge.
You must have a proof of correctness for your method.

CODE:

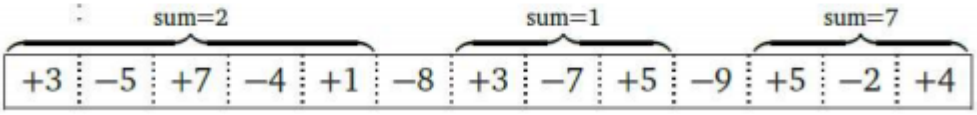
```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n;
    cout<<"Enter number of persons"<<endl;
    cin>>n;
    int person[n];
    set<int> a,b;
    set<int>::iterator it;
    set<int>::reverse_iterator rit;
    cout<<"Enter person's crossing time"<<endl;
    for(int i = 0; i <n;i++)
    {
        cin>>person[i];
        a.insert(person[i]);
    }
    /*for(int i = 0;i<n;i++)
        cout<<person[i]<<"  ";
    cout<<endl;
    for(it=a.begin(); it!=a.end();it++)
        cout<<*it<<"  ";*/
    if(n == 1)
    {
        cout<<"the minimum time required for a single person to cross the bridge is
"<<person[0]<<endl;
        exit(0);
    }
    int first,second;
    int revfirst,revsecond;
    int back;
    while(!a.empty())
    {
        //1st step
        it=a.begin();
        first=*it; second=*(++it);
        cout<<"step ---> "<<first<<" "<<second<<endl;
        b.insert(first); b.insert(second);
        a.erase(first); a.erase(second);
        if(a.empty())
            exit(0);
        it=b.begin();
        back = *it;
        b.erase(back);
        a.insert(back);
        cout<<"step  <--- "<<back<<endl;
```

```
        //2nd step
        if(a.empty())
            exit(0);
        rit=a.rbegin();
        revfirst=*rit; revsecond=*(++rit);
        cout<<"step ---> "<<revfirst<<" "<<revsecond<<endl;
        b.insert(revfirst); b.insert(revsecond);
        a.erase(revfirst); a.erase(revsecond);
        if(a.empty())
            exit(0);
        it=b.begin();
        back = *it;
        b.erase(back);
        a.insert(back);
        cout<<"step  <--- "<<back<<endl;
    }
    return 0;
}
```

This algorithm works correctly because we have made an optimal choise at every step.
Initially, we take two persons with smallest time and they cross the bridge. Then the person with least time out of the two returns. Then we take two persons with largest time and they cross the bridge. Then we take the person with least time and he returns with torch. This strategy is called greedy strategy and hence, it ends up in optimal solution.

Q. 30. Suppose you are given an array A[1 .. n] of integers, each of which may be positive, negative, or zero. A contiguous subarray A[i .. j] is called a positive interval if the sum of its entries is greater than zero. Describe and analyze an algorithm to compute the minimum number of positive intervals that cover every positive entry in A. For example, given the following array as input, your algorithm should output the number 3.



CODE:

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int n,a[1000],c[1000],d[1000],i,ans=1;
    cin>>n;
    for(i=0;i<n;i++)
        cin>>a[i];
    int k=0,r=0;
    for(i=0;i<n;i++)
    {
        if(a[i]>=0)
        {
            if(k==0)
                c[k]=a[i];
            else
                c[k]=c[k-1] +a[i];
            k++;
        }
        else
        {
            if(r==0)
                d[r]=a[i];
            else
                d[r]=d[r-1]+a[i];
            r++;
        }
    }
    r=-1;
    k=-1;
    int y=0;
    int z=0;
    if(a[0]>=0)
        k++;
    else
        r++;

    for(i=1;i<n;i++)
    {
        if(a[i]>=0)
            k++;
        else
            r++;

        if(a[i-1]<0 && a[i]>0)
        {
            int x=c[k]+d[r]-y-z;
            if(x<0)
            {
                y=c[k-1];
                z=d[r];
```

```
                if(k>0)
                ans++;
            }
        }
    }
    cout<<"Minimum No.of Posiitve Intervals :  ";
    cout<<ans<<endl;;
    return 0;
}
```

This algorithm works correctly because we are making an optimal choise at each step.
At every step, we are making the optimal decision and cumulatively, this decision will lead to an optimal result.

Question 30:

Write a program using greedy techniques for yours own two problems and prove the correctness of yours own greedy algorithm.

CODE:

```
/* Algorithm to find minimum number of coins */
#include <bits/stdc++.h>
using namespace std;

int deno[] = {1, 2, 5, 10, 20, 50, 100, 500, 1000};
int n = sizeof(deno)/sizeof(deno[0]);

void findMin(int V)
{
    vector<int> ans;

    for (int i=n-1; i>=0; i--)
    {
        while (V >= deno[i])
        {
           V -= deno[i];
           ans.push_back(deno[i]);
        }
    }
    for (int i = 0; i < ans.size(); i++)
           cout << ans[i] << "  ";
}

int main()
{
   int n = 93;
   cout << "minimal number of change for " << n << " is ";
   findMin(n);
   return 0;
}
```

Question 30:

Write a program using greedy techniques for yours own two problems and prove the correctness of yours own greedy algorithm.

CODE:

```
/* Activity Selection problem */
#include<stdio.h>

void printMaxActivities(int s[], int f[], int n)
{
    int i, j;

    printf ("Following activities are selected \n");
    i = 0;
    printf("%d ", i);

    for (j = 1; j < n; j++)
    {
      if (s[j] >= f[i])
      {
          printf ("%d ", j);
          i = j;
      }
    }
}

int main()
{
    int s[] =  {1, 3, 0, 5, 8, 5};
    int f[] =  {2, 4, 6, 7, 9, 9};
    int n = sizeof(s)/sizeof(s[0]);
    printMaxActivities(s, f, n);
    return 0;
}
```