

```
void bfs()
{
    for(r=0, c=0; r<m; r++)
        nQ(r, c, -1, mat[r][c]);
    do
    {
        dQ(&r, &c, &p, &s);
        if(c<n-1)
            for(i=-1; i<2; i++)
            {
                nr=(m+r+i)%m, nc=c+1;
                if(M[nr][nc] > s+mat[nr][nc])
                    nQ(nr, nc, p, s+mat[nr][nc]), M[nr][nc] = s+mat[nr][nc];
            }
        else if(s<finalSum)
            finalSum = s, leaf = p;
    } while(rear!=front);
}

void dfs(int leaf)
{
    if(Q[leaf][2]==-1)
    {
        printf(" <%d, %d>", Q[leaf][0]+1, Q[leaf][1]+1);
        return;
    }
    dfs(Q[leaf][2]);
    printf(" <%d, %d>", Q[leaf][0]+1, Q[leaf][1]+1);
}

void main()
{
    clrscr();
    int i, j, t;
    init();
    freopen("in.txt", "r", stdin);
    scanf("%d%d", &m, &n);
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
        {
            scanf("%d", &t);
            mat[i][j] = t;
        }
    bfs();

    printf("Final sum: %d\nPath:", finalSum);

    dfs(leaf);
}
```

Floyed Warshal**Input**

```
5 7
1 2 4
1 3 1
1 5 6
2 5 3
2 4 1
3 2 1
4 5 1
0 0
```

Code :

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

#define N 100
#define INF MAXINT

int mat[N][N], path[N][N], n, e;

void initMat()
{
    for(int i=1; i<=n; i++)
        for(int j=1; j<=n; j++)
            mat[i][j] = INF;
}

void initPath()
{
    for(int i=1; i<=n; i++)
        for(int j=1; j<=n; j++)
            if(mat[i][j]!=INF)
                path[i][j] = j;
            else
                path[i][j] = 0;
}

void floyd_warshall()
{
    for(int k=1; k<=n; k++)
        for(int i=1; i<=n; i++)
            for(int j=1; j<=n; j++)
                if(mat[i][k]!=INF && mat[k][j]!=INF)
                    if(mat[i][k]+mat[k][j] < mat[i][j])
                        mat[i][j] = mat[i][k] + mat[k][j],
                        path[i][j] = path[i][k];
}
```

```
}

void showPath(int i, int j)
{
    if(i==j)
    {
        printf("->%d", i);
        return;
    }
    printf("->%d", i);
    showPath(path[i][j], j);
}

void main()
{
    while(scanf("%d%d", &n, &e) && n && e)
    {
        initMat();
        for(int i, j, c, k=0; k<e; k++)
        {
            scanf("%d%d%d", &i, &j, &c);
            mat[i][j] = c;
        }
        initPath();
        floyd_warshall();
        for(i=1; i<=n; i++)
        {
            for(j=1; j<=n; j++)
                if(path[i][j])
                {
                    printf("%d", i);
                    showPath(path[i][j], j);
                    printf("\n");
                }
            printf("\n");
        }
    }
}
```

Graph Coloring

```
#include<stdio.h>
int a[20][20],x[20],n,m;
void next(int k)
{
    int j;
    while(1)
```

```

    {
        x[k]=(x[k]+1)%(m+1);
        if(x[k]==0)
            return;
        for(j=1;j<=n;j++)
            if(a[k][j]!=0&& x[k]==x[j])
                break;
        if(j==n+1)
            return;
    }
}
void mcolor(int k)
{
    int j;
    while(1)
    {
        next(k);
        if(x[k]==0)
            return;
        if(k==n)
        {
            printf("      ");
            for(j=1;j<=n;j++)
                printf("%2d",x[j]);

        }
        else
            mcolor(k+1);
    }
}
void main()
{
    int i,u,v;
    printf("\n\n Enter how many colors : ");
    scanf("%d",&m);
    printf("\n\n Enter how many nodes(0<n<20) :");
    scanf("%d",&n);
    printf("\n\n Enter your edges(ex- u sp v) (press 'e' for end) : \n");
    for(i=1;i<=(n*n)/2;i++)
    {
        if(getchar()=='e')
            break;
        scanf("%d%d",&u,&v);
        a[u][v]=a[v][u]=1;
    }mcolor(1); printf("\n\n");}

```

Cycle Detection (Hamiltonian)

```

#include<stdio.h>
int a[20][20],x[20],n;
void next(int k)
{
    int j;

```

```
while(1)
{
    x[k]=(x[k]+1)%(n+1);
    if(x[k]==0)
        return;
    if((a[x[k-1]][x[k]])!=0)
    {
        for(j=1;j<=k-1;j++)
            if(x[k]==x[j])
                break;
        if(j==k)
            if((k<n)|| (k==n&&a[x[n]][x[1]]!=0))
                return;
    }
}
}
void hamilt(int k)
{
    int j;
    while(1)
    {
        next(k);
        if(x[k]==0)
            return;
        if(k==n)
        {
            printf("      ");
            for(j=1;j<=n;j++)
                printf("%2d",x[j]);
        }
        else
            hamilt(k+1);
    }
}
void main()
{
    int i,u,v;
    x[1]=1;
    printf("\n\n Enter how many nodes (0<n<20) :");
    scanf("%d",&n);
    printf("\n\n Enter your edges(ex- u sp v) (press 'e' for end) : \n");
    for(i=1;i<=(n*n)/2;i++)
    {
        if(getchar()=='e')
            break;
        scanf("%d%d",&u,&v);
        a[u][v]=a[v][u]=1;
    }
    hamilt(2);
    printf("\n\n");
}
```

Finding Articulation Point

```

#include<stdio.h>
int d[11],num=1,b[11][11],l[11],at[11],s=1;
void art(int u,int v)
{
    int i,j=1,w,f=0;
    d[u]=num;
    l[u]=num;
    num++;
    for(i=1;b[u][i]!=0;i++)
    {
        w=b[u][i];
        if(d[w]==0)
        {
            art(w,u);
            l[u]=(l[u]<l[w])?l[u]:l[w];
        }
        else if(w!=v)
            l[u]=(l[u]<d[w])?l[u]:d[w];
        if(d[u]<=l[w])
            f=1;
    }
    if(f)
        at[s++]=u;
}
void main()
{
    int i,j,a[11][11],n,u,v,k,f=0;
    for(i=1;i<11;i++)
        for(j=1;j<11;j++)
            a[i][j]=0;
    printf("\n\n Enter how many nodes(0<n<11) :");
    scanf("%d",&n);
    printf("\n\n Enter your edges(ex- u sp v) (press 'e' for end) : ");
    for(i=1;i<=(n*n)/2;i++)
    {
        if(getchar()=='e')
            break;
        scanf("%d%d",&u,&v);
        a[u][v]=a[v][u]=1;
    }
    for(i=1;i<=n;i++)
    {
        k=1;
        for(j=1;j<=n;j++)
            if(a[i][j])
            {
                b[i][k]=j;
                k++;
            }
    }
}

```

```
        b[i][k]=0;
    }
    for(j=1,i=1;b[l][j]!=0;j++)
    {
        k=b[l][j];
        if(b[k][2])
            i++;
    }
    if(j==i)
        f=1;
    art(1,1);
    at[s]=-9;
    printf("\n\n Articulation points are : ");
    if(!f)
        for(i=1;at[i]!=-9;i++)
            printf("%3d",at[i]);
    if(f)
        for(i=1;at[i]!=1;i++)
            printf("%3d",at[i]);
    printf("\n\n");
}
```

APPENDIX C

STANDARD TEMPLATE LIBRARY (STL)



The programming languages for the contest will be C, C++, Pascal, and Java and the `scanf()`/`printf()` family of functions. The C++ string library and Standard Template Library (STL) is also available. This part of this book contains an introductory note about STL^[9].

Standard Template Library (STL)

The Standard Template Library, or *STL*, is a C++ library of container classes, algorithms, and iterators; it provides many of the basic algorithms and data structures of computer science. The STL is a *generic* library, meaning that its components are heavily parameterized: almost every component in the STL is a template. You should make sure that you understand how templates work in C++ before you use the STL.

Containers and algorithms

Like many class libraries, the STL includes *container* classes: classes whose purpose is to contain other objects. The STL includes the classes `vector`, `list`, `deque`, `set`, `multiset`, `map`, `multimap`, `hash_set`, `hash_multiset`, `hash_map`, and `hash_multimap`. Each of these classes is a template, and can be instantiated to contain any type of object. You can, for example, use a `vector<int>` in much the same way as you would use an ordinary C array, except that `vector` eliminates the chore of managing dynamic memory allocation by hand.

```
vector<int> v(3);           // Declare a vector of 3 elements.
    v[0] = 7;
    v[1] = v[0] + 3;
    v[2] = v[0] + v[1];     // v[0] == 7, v[1] == 10, v[2] == 17
```

The STL also includes a large collection of *algorithms* that manipulate the data stored in containers. You can reverse the order of elements in a vector, for example, by using the `reverse` algorithm.

```
reverse(v.begin(), v.end()); // v[0] == 17, v[1] == 10, v[2] == 7
```

There are two important points to notice about this call to `reverse`. First, it is a global function, not a member function. Second, it takes two arguments rather than one: it operates on a *range* of elements, rather than on a container. In this particular case the range happens to be the entire container `v`.

The reason for both of these facts is the same: `reverse`, like other STL algorithms, is decoupled from the STL container classes.

```
double A[6] = { 1.2, 1.3, 1.4, 1.5, 1.6, 1.7 };
    reverse(A, A + 6);
    for (int i = 0; i < 6; ++i)
        cout << "A[" << i << "] = " << A[i];
```

This example uses a *range*, just like the example of reversing a vector: the first argument to `reverse` is a pointer to the beginning of the range, and the second argument points one

element past the end of the range. This range is denoted $[A, A + 6)$; the asymmetrical notation is a reminder that the two endpoints are different, that the first is the beginning of the range and the second is *one past* the end of the range.

Website for downloading STL

<http://www.sgi.com/tech/stl/download.htm>

Individual files in STL

algo.h	hash_map.h	numeric	stdexcept	stl_heap.h	stl_slist.h
algorith	hash_set	pair.h	stl_algo.h	stl_iterator.h	stl_stack.h
algorithm	hash_set.h	pthread_alloc	stl_algorith	stl_iterator_base.h	stl_string_fwd.h
alloc.h	hashtable.h	pthread_alloc.h	stl_alloc.h	stl_list.h	stl_tempbuf.h
bitset	heap.h	queue	stl_bvector.h	stl_map.h	stl_threads.h
bvector.h	iterator	rope	stl_config.h	stl_multimap.h	stl_tree.h
char_traits.h	iterator.h	rope.h	stl_construct.h	stl_multiset.h	stl_uninitialized.h
concept_checks.h	limits	ropeimpl.h	stl_traits_fns.h	stl_numeric.h	stl_vector.h
container_concepts.h	list	sequence_concepts.h	stl_deque.h	stl_pair.h	string
defalloc.h	list.h	set	stl_exception.h	stl_queue.h	tempbuf.h
deque	map	set.h	stl_function.h	stl_range_error.h	tree.h
deque.h	map.h	slist	stl_hash_fun.h	stl_raw_storage_iter.h	type_traits.h
function.h	memory	slist.h	stl_hash_map.h	stl_relops.h	utility
functional	multimap.h	stack	stl_hash_set.h	stl_rope.h	valarray
hash_map	multiset.h	stack.h	stl_hashtable.h	stl_set.h	vector

Which compilers are supported?

The STL has been tested on these compilers: SGI 7.1 and later, or 7.0 with the -n32 or -64 flag; gcc 2.8 or egcs 1.x; Microsoft 5.0 and later. (But see below.) Boris Fomitchev distributes a port for some other compilers.

If you succeed in using the SGI STL with some other compiler, please let us know, and please tell us what modifications (if any) you had to make. We expect that most of the changes will be restricted to the `<stl_config.h>` header.

STL EXAMPLES**Search**

```
const char S1[] = "Hello, world!";
const char S2[] = "world";
const int N1 = sizeof(S1) - 1;
const int N2 = sizeof(S2) - 1;

const char* p = search(S1, S1 + N1, S2, S2 + N2);
printf("Found subsequence \"%s\" at character %d of sequence
\"%s\".\n",
      S2, p - S1, S1);
```

Queue

```
int main() {
    queue<int> Q;
    Q.push(8);
    Q.push(7);
    Q.push(6);
    Q.push(2);

    assert(Q.size() == 4);
    assert(Q.back() == 2);

    assert(Q.front() == 8);
    Q.pop();

    assert(Q.front() == 7);
    Q.pop();

    assert(Q.front() == 6);
    Q.pop();
    assert(Q.front() == 2);
    Q.pop();
    assert(Q.empty());}
```

Doubly linked list

```
list<int> L;
L.push_back(0);
L.push_front(1);
L.insert(++L.begin(), 2);
copy(L.begin(), L.end(), ostream_iterator<int>(cout, " "));
// The values that are printed are 1 2 0
```

Sort

```
int A[] = {1, 4, 2, 8, 5, 7};
const int N = sizeof(A) / sizeof(int);
sort(A, A + N);
copy(A, A + N, ostream_iterator<int>(cout, " "));
// The output is " 1 2 4 5 7 8".
```

Complexity

$O(N \log(N))$ comparisons (both average and worst-case), where N is last - first.

Binary Search

```
int main()
{
    int A[] = { 1, 2, 3, 3, 3, 5, 8 };
    const int N = sizeof(A) / sizeof(int);

    for (int i = 1; i <= 10; ++i) {
        cout << "Searching for " << i << ": "
              << (binary_search(A, A + N, i) ? "present" : "not present") <<
endl;
    }
}
```

The output

```
Searching for 1: present
Searching for 2: present
Searching for 3: present
Searching for 4: not present
Searching for 5: present
Searching for 6: not present
Searching for 7: not present
Searching for 8: present
Searching for 9: not present
Searching for 10: not present
```

APPENDIX D

PC² CONTEST ADMINISTRATION AND TEAM GUIDE



PC² is a dynamic, distributed real-time system designed to manage and control Programming Contests. It includes support for multi-site contests, heterogeneous platform operations including mixed Windows and Unix in a single contest, and dynamic real-time updates of contest status and standings to all sites. Here we describe the steps required to install, configure, and run a contest using PC². Further information on PC², including how to obtain a copy of the system, can be found at <http://www.ecs.csus.edu/pc2>.

Programming Contest Judge Software - PC²

PC² operates using a client-server architecture. Each site in a contest runs a single PC² *server*, and also runs multiple PC² *clients* which communicate with the site server. Logging into a client using one of several different types of PC² accounts (Administrator, Team, Judge, or Scoreboard) enables that client to perform common contest operations associated with the account type, such as contest configuration and control (Administrator), submitting contestant programs (Team), judging submissions (Judge), and maintaining the current contest standings (Scoreboard).

PC² clients communicate only with the server at their site, regardless of the number of sites in the contest. In a multi-site contest, site servers communicate not only with their own clients but also with other site servers, in order to keep track of global contest state. The following communication requirements must therefore be met in order to run a contest using PC²: (1) a machine running a PC² server must be able to communicate via TCP/IP with every machine running a PC² client at its site; and (2) in a multi-site contest, every machine running a PC² server must be able to communicate via TCP/IP with the machines running PC² servers at every other site^[1]. In particular, there must not be any firewalls which prohibit these communication paths; the system will not operate if this communication is blocked. It is not necessary for client machines to be able to contact machines at other sites.

Each PC² module (server or client) reads one or more .ini initialization files when it starts; these files are used to configure the module at startup. The client module also tailors its configuration when a user (Team, Judge, etc.) logs in. In a typical PC² contest configuration, each Team, Judge, etc. uses a separate physical machine, and each of these machines runs exactly one client module. It is possible to have multiple clients running on the same physical machine, for example by having different users logging in to different accounts on a shared machine. In this case, each user (Team, Judge, etc.) will be executing their own Java Virtual Machine (JVM), and must have their own separate directory structure including their own separate copy of the PC² initialization files in their account.

PC² Setup for Administrator

For those people who hate to read manuals and would rather take a chance with a shortcut list, here is a *very terse* summary of the steps necessary to install PC² and get it running. Please note that this is provided as a convenience for geniuses (or gluttons for punishment). The remainder of the manual was written to help everyone else. If you have

* For further information about PC² please check : <http://www.ecs.csus.edu/pc2>

problems after following this list, *please read the rest of the manual* before sending us a request for help.

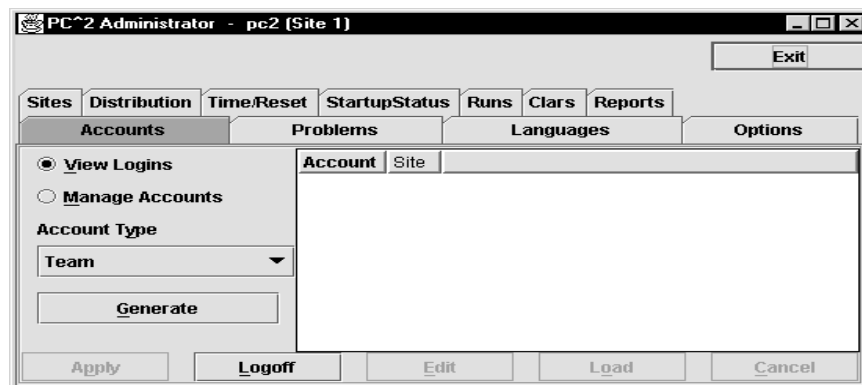
- ❑ Install Java (version 1.3.1 or greater) ;
- ❑ Install PC² by unzipping the PC² distribution to the PC² installation directory;
- ❑ Add the Java *bin* directory and the PC² installation directory to the PATH;
- ❑ Add java/lib, and the PC² installation directory to the CLASSPATH;
- ❑ Modify the *sitelist.ini* file as necessary to specify each site server name.
- ❑ Edit the *pc2v8.ini* file to point servers and clients to the server IP:port and to specify the appropriate site server name; put the modified .ini file on every server and client machine;

```
# sample pc2v8.ini file for site named Site

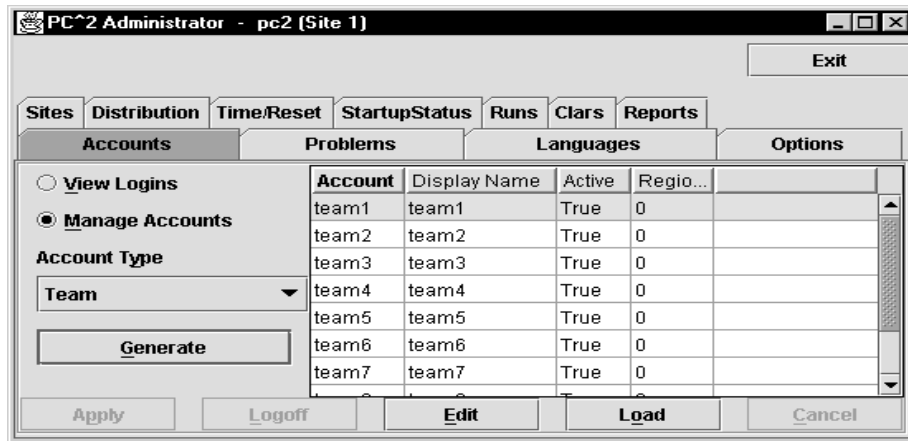
[client]
# tell the client what site it belongs to
# and where to find its server (IP and port)
site=Site1
server=192.168.1.117:50002

[server]
# tell the server which site it is serving
site=Site1
```

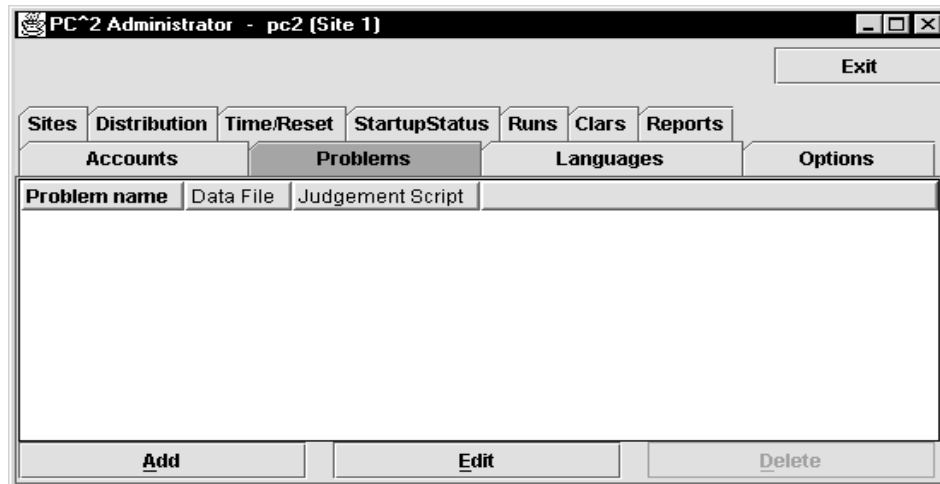
- ❑ Start a PC² server using the command pc2server and answer the prompted question.
- ❑ Start a PC² Admin client using the command pc2admin and login using the name root and password root.



- ❑ Configure at least the following contest items via the Admin:
- ❑ Accounts (generate the necessary accounts);

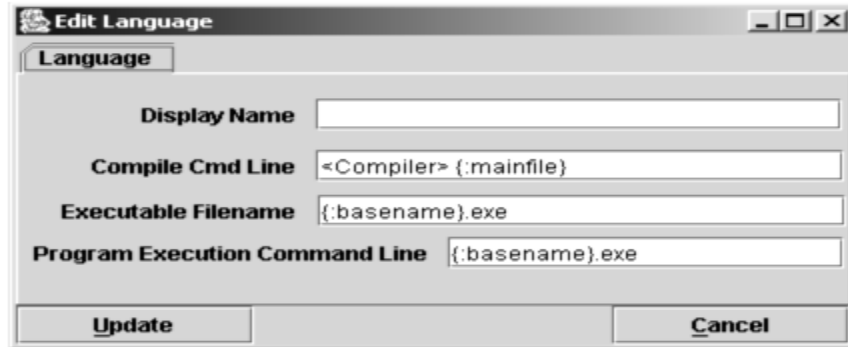


- ❑ Problems (create one or more contest problems, specifying the problem input data file if there is one);



- ❑ Languages (create contest languages, specifying the language name, compile command line, executable filename, and execution command line).

❑ Language Parameter for Turbo C/C++ : `tcc -Ic:\tc3\bin -Lc:\tc3\lib {;mainfile}`

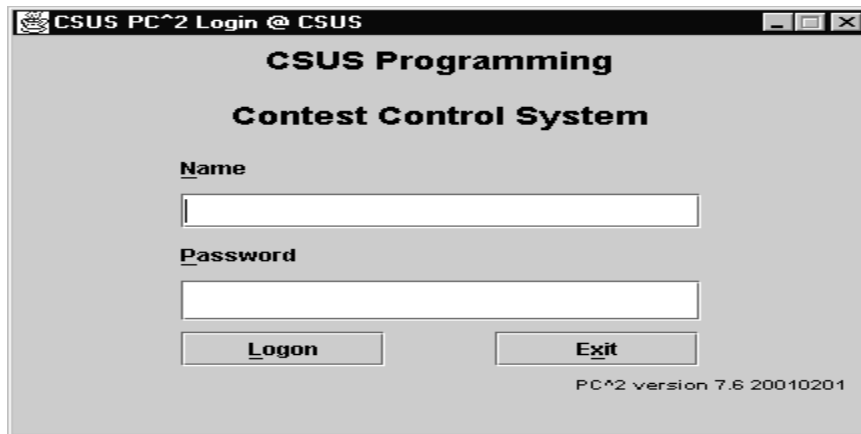


- ❑ Press the Start Contest button on the Admin Time/Reset tab;
- ❑ Start a PC² client on each Team and Judge machine and log in using the Admin-created accounts and passwords.
- ❑ Start a PC² client on the Scoreboard machine and log in using the board1 Scoreboard account/password; arrange for the scoreboard-generated HTML files to be accessible to users browsers.



PC² Team Guide (For Contestants)

This guide is intended to familiarize you with the process of submitting programs to Contest Judges using the PC² (“P-C-Squared”) **P**rogramming **C**ontest **C**ontrol system. Starting PC² will bring up the PC² **login screen**, shown below:



CSUS PC² Login @ CSUS

**CSUS Programming
Contest Control System**

Name

Password

Logon **Exit**

PC² version 7.6 20010201

To login to PC², click once on the **Name** box on the login screen, enter your assigned team ID, press the TAB key or click on the **Password** box, then enter your assigned password. Your team ID will be of the form **teamxx**, where xx is your assigned team number (for example, “**team3**” or “**team12**”). After entering your team name and password, click on the **Logon** button.

Submitting a Program to the Judges

Once the system accepts your login, you will be at the PC² **Main Menu** screen, shown below. Note that the team ID (“team1” in this case) and the team’s site location (“CSUS” in this case) are displayed in the title bar.



PC² - team1@CSUS

Exit

Submit **Clarifications** **Runs** **Settings**

Problem **Bowling**

Language **Java**

Main File **Select** C:\work\bowling.java

Test

Submit

Additional Files

Add **Remove**

Clicking on the **SUBMIT** tab near the top of the screen displays the **Submit Run** screen, which is shown above.

Clicking in the **Problem** field will display a list of the contest problems; choose the problem for which you wish to submit a program (“run”) to the Judges (in the example, a problem named “Bowling” has been chosen).

Clicking in the **Language** field will display a list of the programming languages allowed in the contest; choose the language used by the program that you wish to submit to the Judges (in the example, “Java” has been chosen).

To submit a program to the Judges, you must specify the name of the file containing your **main program**. Click on the **Select** button to invoke the “File Dialog” which lets you locate and select your main file. The Dialog lets you automatically navigate to the correct path and file location (in the example, the main program file “C:\work\bowling.java” has been selected).

If your program consists of more than one file, you must enter the additional file names in the **Additional Files** box. Click the **Add** button to invoke the dialog which lets you locate and select your additional files; select a file in the box and click the **Remove** button to remove a previously-selected file.

Important: you *must* specify the name of your *main program* file in the **Main File** field, *not* in the **Additional Files** box! Select only source code files for submission to the Judges. Do not submit data files or executable files.

PC² Uninstall

To uninstall PC² it is only necessary to undo the above installation steps; that is, remove the \$PC2HOME directory and its contents and restore the system environment variables to their former values . PC² itself does not make any changes to any machine locations outside those listed above either during installation or execution. In particular, for example, it makes no entries in the registry in a Windows environment, nor does it copy any files to locations outside the installation directories in any environment.

APPENDIX E

IMPORTANT WEBSITES/ FOR ACM/ICPC PROGRAMMERS



Now a days the world is open for learners and the Internet makes the world inside our room. There are many websites now on the Internet for the beginners as well as programmers. Some sites are for contest and tutorial, some for books, some sites are for programming language. Mathematics is one of the essential for programming and there are a lot of mathematical site in the net. The pioneers also publish their ideas and advice on the Internet for us. In this page you will find the link of those pages.^[10]