D. Source Codes of Several Self-developed Programs

Calculate RMSD from two different poses of the same molecule in the PDB format. The general synopsis for using is:

rmsd [pose01.pdb] [pose02.pdb]

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main(int argc, char *argv[])
{
  char buffera[100];
  char bufferb[100];
  char coordinatex[8];
  char coordinatey[8];
  char coordinatez[8];
  float coordinatea[100][3];  //record atom coordinates of ligA
  float coordinateb[100][3];  //record atom coordinates of ligB
  char head[6];  //the head of the PDB format
  int i = 0;  //index
  int atomnuma = 0;  //atom number of ligA
  int atomnumb = 0;  //atom number of ligB
  float sum= 0.0;
  float sumsquare = 0.0;
  float rmsd = 0.0;
  FILE *liga;
  FILE *ligb;
  liga = fopen (argv[1], "r");
  ligb = fopen (argv[2], "r");
  if ( liga == NULL )
  {
     printf ("Can't open the file %s\n", argv[1]);
     exit(1);
  }
  if ( ligb == NULL )
  {
     printf ("Can't open the file %s\n", argv[2]);
     exit (1);
  }
  while ( (fgets(buffera, 100, liga)) != NULL)
  {
     strncpy (head, &buffera[0], 6);
     if ( strncmp (head, “HETATM”, 6) == 0 ||
```

```
          strncmp (head, “ATOM”, 6) == 0 )
      {
         strncpy (coordinatex, &buffera[30], 8);
         strncpy (coordinatey, &buffera[38], 8);
         strncpy (coordinatez, &buffera[46], 8);
         coordinatea[atomnuma][0] = atof(coordinatex);
         coordinatea[atomnuma][1] = atof(coordinatey);
         coordinatea[atomnuma][2] = atof(coordinatez);
         atomnuma++;
      }
   }
   while ( (fgets(bufferb, 100, ligb)) != NULL)
   {
      strncpy (head, &bufferb[0], 6);
      if ( strncmp (head, "HETATM", 6) == 0 ||
          strncmp (head, “ATOM  ”, 6) == 0 )
      {
         strncpy (coordinatex, &bufferb[30], 8);
         strncpy (coordinatey, &bufferb[38], 8);
         strncpy (coordinatez, &bufferb[46], 8);
         coordinateb[atomnumb][0] = atof(coordinatex);
         coordinateb[atomnumb][1] = atof(coordinatey);
         coordinateb[atomnumb][2] = atof(coordinatez);
         atomnumb++;
      }
   }
   for (i=0;i<atomnuma;i++)
   {
      sumsquare=
      ( (coordinatea[i][0] - coordinateb[i][0])*
      (coordinatea[i][0] - coordinateb[i][0]) )+
      ( (coordinatea[i][1] - coordinateb[i][1])*
      (coordinatea[i][1] - coordinateb[i][1]) )+
      ( (coordinatea[i][2] - coordinateb[i][2])*
      (coordinatea[i][2] - coordinateb[i][2]) );
      sum += sumsquare;
   }
   rmsd = sqrt(sum/atomnuma);
   printf("RMSD = %8.3f\n", rmsd);
   fclose (liga);
   fclose (ligb);
   return 0;
}
```

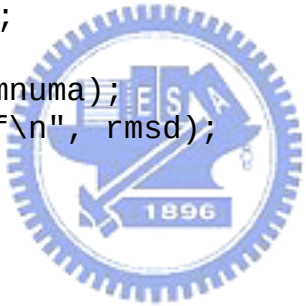
Calculate RMSD from two different poses of the same molecule in the mol2 format. The general synopsis for using is:

rmsd_mol2 [pose01.mol2] [pose02.mol2]

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
int main(int argc, char *argv[])
{
  char buffera[120];
  char bufferb[120];
  char coordinatex[10];
  char coordinatey[10];
  char coordinatez[10];
  float coordinatea[100][3];
  float coordinateb[100][3];
  char seps[] = " ";  //separate character
  char *token;
  int i = 0;  //index
  int atomnuma = 0;
  int atomnumb = 0;
  int tokennum = 0;
  float sum= 0.0;
  float sumsquare = 0.0;
  float rmsd = 0.0;
  FILE *liga;
  FILE *ligb;
  liga = fopen (argv[1], "r");
  ligb = fopen (argv[2], "r");
  if ( (argv[1] == NULL) || (argv[2] == NULL) )
  {
     printf ("format: rmsd_mol2 file1.mol2 file2.mol2\n");
     printf ("Please input file1.mol2 and file2.mol2.\n");
     exit (1);
  }
  if ( liga == NULL )
  {
     printf ("Can't open the file %s\n", argv[1]);
     exit (1);
  }
  if ( ligb == NULL )
  {
     printf ("Can't open the file %s\n", argv[2]);
```

```
    exit (1);
}
while ( fgets(buffera, 120, liga) != NULL )
{
    if ( !strncmp(buffera, "@<TRIPOS>ATOM", 13) )
    {
        while ( strncmp(fgets(buffera, 120, liga),
                "@<TRIPOS>BOND", 13) )
        {
            if ( (buffera[8] != 'H') && (buffera[8] != '*') )
            {
                tokennum = 0;
                token = strtok ( buffera, seps );
                while ( token != NULL )
                {
                    if ( tokennum == 2 )
                        strcpy (coordinatex, token);
                    else if ( tokennum == 3 )
                        strcpy (coordinatey, token);
                    else if ( tokennum == 4 )
                        strcpy (coordinatez, token);
                    token = strtok( NULL, seps );
                    tokennum++;
                }
                coordinatea[atomnuma][0] = atof(coordinatex);
                coordinatea[atomnuma][1] = atof(coordinatey);
                coordinatea[atomnuma][2] = atof(coordinatez);
                atomnuma++;
            }
        }
    }
}
while ( fgets(bufferb, 120, ligb) != NULL )
{
    if ( !strncmp(&bufferb[0], "@<TRIPOS>ATOM", 13) )
    {
        while ( strncmp( fgets(bufferb, 120, ligb),
                  "@<TRIPOS>BOND", 13 ) )
        {
            if ( (bufferb[8] != 'H') && (bufferb[8] != '*') )
            {
                tokennum = 0;
                token = strtok ( bufferb, seps );
                while ( token != NULL )
                {
                    if ( tokennum == 2 )
                        strcpy (coordinatex, token);
                    else if ( tokennum == 3 )
                        strcpy (coordinatey, token);
                    else if ( tokennum == 4 )
```

```
                    strcpy (coordinatez, token);
                token = strtok( NULL, seps );
                tokennum++;
            }
            coordinateb[atomnumb][0] = atof(coordinatex);
            coordinateb[atomnumb][1] = atof(coordinatey);
            coordinateb[atomnumb][2] = atof(coordinatez);
            atomnumb++;
        }
      }
    }
  }
  for ( i=0 ; i<atomnuma ; i++ )
  {
     sumsquare=
     ( (coordinatea[i][0]-coordinateb[i][0])*
     (coordinatea[i][0]-coordinateb[i][0]) )+
     ( (coordinatea[i][1]-coordinateb[i][1])*
     (coordinatea[i][1]-coordinateb[i][1]) )+
     ( (coordinatea[i][2] - coordinateb[i][2])*
     (coordinatea[i][2] - coordinateb[i][2]) );
     sum += sumsquare;
  }
  rmsd = sqrt(sum/atomnuma);
  printf("RMSD = %8.3f\n", rmsd);
  fclose (liga);
  fclose (ligb);
  return 0;
}
```



Count the number of hydrogen bonds between docked poses and the protein from a list of docked poses (docklog.txt) and the cavity file. The general synopsis for using is:

readhbond [cavity.pdb] [outfile.txt]

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

#define maxresnum 200

int main(int argc, char *argv[])
{
  char buffer[100];
  char ligbuffer[100];  //buffer of the ligand file
  char listbuffer[100];  //buffer of the list file
  char path[50];  //the path of the local disk
  char temp[5];
  //temporary buffer of the residue number from the cavity file
  char tempresnum[5];
  //temporary buffer of the residue number from the docked ligand
  int len;  //length
  int i = 0;  //index
  int rescount = 0;  //the count of residues
  int TEMPRESNUM = 0;
  struct CAVITY
  {
     char residue[4];
     int resnum;
     int hbondcount;
  }cavity[maxresnum];
  for (i=0; i<maxresnum; i++)
     cavity[i].hbondcount = 0;
  FILE *cav;
  FILE *docklog;
  FILE *ligand;
  FILE *output;
  cav = fopen (argv[1], "r");
  if (argv[1] == NULL || argv[2] == NULL)
  {
     printf ("To count the number of Hbond formed
              between residue and ligand.\n");
     printf ("format: readhbond cavity.pdb output\n");
     exit (1);
  }
```

```
if (cav == NULL)
{
   printf ("Can't open the file %s", argv[1]);
   exit (1);
}
output = fopen (argv[2], "a");
if (output == NULL)
{
   printf ("Can't open the file %s", argv[2]);
   exit (1);
}
while (fgets(buffer, 100, cav) != NULL)
{
   if ( (!strncmp (buffer, "ATOM  ", 6)) ||
        (!strncmp (buffer, "HETATM", 6)))
   {
      strncpy (temp, &buffer[22], 4);
      if (rescount == 0)  //record the first residue
      {
         cavity[rescount].resnum = atoi(temp);
         strncpy (cavity[rescount].residue, &buffer[17], 3);
         cavity[rescount].residue[3] = '\0';
         rescount++;
      }
      else
      {
         if (cavity[rescount-1].resnum != atoi (temp))
         {
            cavity[rescount].resnum = atoi (temp);
            strncpy (cavity[rescount].residue,
                     &buffer[17], 3);
            cavity[rescount].residue[3] = '\0';
            rescount++;
         }
      }
   }
}
fclose(cav);
docklog = fopen ("docklog.txt", "r");
if (docklog == NULL)
{
   printf ("Please prepare the ligand file list docklog.txt!\n");
   exit (1);
}
while (fgets (listbuffer, 100, docklog) != NULL)
{
     len = strlen( listbuffer );
     strncpy(path, &listbuffer[0], len - 1 );
     path[ len - 1 ] = '\0';
     printf("%s\n",path);
```

```
        ligand = fopen (path, "r");
        if (ligand == NULL)
        {
           printf ("Can't open the ligand file %s\n", path);
           exit (1);
        }
        while (fgets (ligbuffer, 100, ligand) != NULL)
        {
           if (!strncmp (ligbuffer, "DHBOND", 6))
           {
              if (ligbuffer[7] == 'P')
              {
                 strncpy (tempresnum, &ligbuffer[8], 4);
                 TEMPRESNUM = atoi(tempresnum);
                 for (i=0; i<rescount; i++)
                 {
                    if (cavity[i].resnum == TEMPRESNUM)
                       cavity[i].hbondcount =
                       cavity[i].hbondcount + 1;
                 }
              }
           }
        }
  }
  fclose (docklog);
  fclose (ligand);
  for (i=0; i<rescount; i++)
  {
     fprintf (output, "%s%d %d\n", cavity[i].residue,
              cavity[i].resnum, cavity[i].hbondcount);
  }
  fclose (output);
  system("PAUSE");
  return 0;
}
```

Generate cavity file for GOLD from the cavity file in the PDB format and the target protein in the mol2 format. The general synopsis for using is:

GenGcav [cavity.pdb][protein.mol2]

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
char seps[] = " ";
char *token;
int tokennum = 0;

void Readmol2( char ResNuma[], char mol2file[] )
{
  char bufferb[120];
  char AtomNum[5];
  char ResNumb[7];
  int atomnumb = 0;
  FILE *ligb;
  ligb = fopen (mol2file, "r");
  if ( ligb == NULL )
  {
     printf ("Can't open the file %s\n", mol2file);
     exit (1);
  }
  FILE *output;
  output = fopen ("cavity.atom", "a");
  if ( output == NULL )
  {
     printf ("Can't open the file cavity.atom to output.\n");
     exit (1);
  }
  while ( fgets(bufferb, 120, ligb) != NULL )
  {
     if ( !strncmp(&bufferb[0], "@<TRIPOS>ATOM", 13) )
     {
        while ( strncmp( fgets(bufferb, 120, ligb),
                "@<TRIPOS>BOND", 13 ) )
        {
           tokennum = 0;
           token = strtok ( bufferb, seps );
           while ( token != NULL )
           {
              if ( tokennum == 0 )
```

```
                strcpy (AtomNum, token);
            else if ( tokennum == 7 )
                strcpy (ResNumb, token);
            token = strtok( NULL, seps );
            tokennum++;
          }
          if ( !strncmp (ResNuma, ResNumb, 6) )
            fprintf (output, "%s\n", AtomNum);
          atomnumb++;
        }
      }
  }
  fclose (ligb);
  fclose (output);
}

int main(int argc, char *argv[])
{
  char buffera[100];
  char ResNum[7];
  char PreResNum[7];
  int atomnuma = 0;
  FILE *liga;
  liga = fopen (argv[1], "r");
  if ( (argv[1] == NULL) || (argv[2] == NULL) )
  {
     printf ("format: GenGcav file1.pdb file2.mol2\n");
     printf ("Please input file1.pdb and file2.mol2.\n");
     exit (1);
  }
  if ( liga == NULL )
  {
     printf ("Can't open the file %s\n", argv[1]);
     exit (1);
  }
  while ( fgets(buffera, 100, liga) != NULL )
  {
     if ( !strncmp(buffera, "ATOM  ", 6) ||
          !strncmp(buffera, "HETATM", 6) )
     {
        tokennum = 0;
        token = strtok ( buffera, seps );
        buffera[21] = 'A';
        while ( token != NULL )
        {
          if ( tokennum == 3 )
            strcpy (ResNum, token);
          else if ( tokennum == 5 )
            strcpy (&ResNum[3], token);
          token = strtok( NULL, seps );
```

```
            tokennum++;
         }
         atomnuma++;
         if ( strcmp (PreResNum, ResNum) )
            Readmol2( ResNum, argv[2] );
         strcpy (PreResNum, ResNum);
      }

  }
  fclose (liga);
  return 0;
}
```