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

#define NumberOfNodes 22

#define NilValue -1

/*??????????? ??? ????? ListElementType ???? ?? ?????????
??? ?? ??? ??? ????? ??? ?????? ??? ???????? ??????? ????
?????????? InitializeStoragePool,ReleaseNode,Insert,TraverseLinked
??? ?? ??????? ??? ?? ??? ????????.*/
typedef struct{
    int AM;
    float grade;
} ListElementType;

typedef int ListPointer;

typedef struct {
    ListElementType Data;
    ListPointer  Next;
} NodeType;

typedef enum {
    FALSE, TRUE
} boolean;

void InitializeStoragePool(NodeType Node[], ListPointer *FreePtr);
void CreateLList(ListPointer *List);
boolean EmptyLList(ListPointer List);
boolean FullLList(ListPointer FreePtr);
void GetNode(ListPointer *P, ListPointer *FreePtr, NodeType Node[]);
void ReleaseNode(NodeType Node[NumberOfNodes], ListPointer P, ListPointer *FreePtr);
void Insert(ListPointer *List, NodeType Node[],ListPointer *FreePtr, ListPointer PredPtr, ListElementType
Item);
void Delete(ListPointer *List, NodeType Node[], ListPointer *FreePtr, ListPointer PredPtr);
void TraverseLinked(ListPointer List, NodeType Node[]);

int main()
{
    int number_of_students,i,j;
    ListPointer AList;
    NodeType Node[NumberOfNodes];
    ListPointer FreePtr,PredPtr;
    ListElementType Item;

    /*??????????? ????? ??.(??????? i)*/
    InitializeStoragePool(Node, &FreePtr);
    CreateLList(&AList);

    /*???????? ??????? ??????? ??? ??????? ???????????.(??????? ii)*/
    do{
        printf("DWSE ARI8MO MA8ITWN:");
        scanf("%d",&number_of_students);
        if(number_of_students < 0 || number_of_students >20)
            printf("MH EPITREPTOS ARI8MOS.PROSPA8ISTE KSANA.\n");
    }while(number_of_students < 0 || number_of_students >20);

    /*???????? ????????? ??????? ??? ???????? ?? ?? ??? ?????????
    TraverseLinked.(??????? iii)*/
    for(i=0; i < number_of_students; i++)
    {
        printf("DWSE ARI8MO MHTRWOU GIA EISAGWGH STH LISTA: ");
        scanf("%d",&Item.AM);

        printf("DWSE BA8MO GIA EISAGWGH STH LISTA: ");
```

```
66          scanf("%f",&Item.grade);
67
68          printf("DWSE TH 8ESH META THN OPOIA 8A GINEI H EISAGWGH STOIXEIOU: ");
69          scanf("%d",&PredPtr);
70          printf("\n");
71
72          printf("Plithos stoixeiwn sth lista %d\n",i+1);
73          Insert(&AList,Node,&FreePtr,PredPtr,Item);
74
75          TraverseLinked(AList,Node);
76      }
77
78      /*???????? ???? ?????? ??? ???????? ??? ??.(??????? iv)*/
79      printf("DWSE TH 8ESH TOY PROHGOUMENOY STOIXEIOY GIA DIAGRAFI: ");
80      scanf("%d",&PredPtr);
81      printf("\n");
82      Delete(&AList,Node,&FreePtr,PredPtr);
83
84      printf("Plithos stoixeiwn sth lista %d\n",i-1);
85      TraverseLinked(AList,Node);
86
87      /*???????? ????????? ??? ???? ??????? ??? ???????? ??.(??????? v)*/
88      for(j=0; j < 2; j++)
89      {
90          printf("DWSE ARI8MO MHTRWOU GIA EISAGWGH STH LISTA: ");
91          scanf("%d",&Item.AM);
92
93          printf("DWSE BA8MO GIA EISAGWGH STH LISTA: ");
94          scanf("%f",&Item.grade);
95
96          printf("DWSE TH 8ESH META THN OPOIA 8A GINEI H EISAGWGH STOIXEIOU: ");
97          scanf("%d",&PredPtr);
98          printf("\n");
99
100         printf("Plithos stoixeiwn sth lista %d\n",i+j);
101         Insert(&AList,Node,&FreePtr,PredPtr,Item);
102
103         TraverseLinked(AList,Node);
104     }
105
106     return 0;
107 }
108
109 void InitializeStoragePool(NodeType Node[], ListPointer *FreePtr)
110 {
111    int i;
112
113     for (i=0; i<NumberOfNodes-1;i++)
114     {
115         Node[i].Next=i+1;
116         Node[i].Data.AM=-1;
117         Node[i].Data.grade=-1;
118
119     }
120     Node[NumberOfNodes-1].Next=NilValue;
121     Node[NumberOfNodes-1].Data.AM=NilValue;
122     Node[NumberOfNodes-1].Data.grade=NilValue;
123     *FreePtr=0;
124 }
125
126 void CreateLList(ListPointer *List)
127 {
128   *List=NilValue;
129 }
130
131 boolean EmptyLList(ListPointer List)
```

```
{
  return (List==NilValue);
}

boolean FullLList(ListPointer FreePtr)
{
  return (FreePtr == NilValue);
}

void GetNode(ListPointer *P, ListPointer *FreePtr, NodeType Node[])
{
  *P = *FreePtr;
  if (!FullLList(*FreePtr))
    *FreePtr =Node[*FreePtr].Next;
}

void ReleaseNode(NodeType Node[], ListPointer P, ListPointer *FreePtr)
{
  Node[P].Next =*FreePtr;
  Node[P].Data.AM = -1;
  Node[P].Data.grade = -1;

  *FreePtr =P;
}

void Insert(ListPointer *List, NodeType Node[],ListPointer *FreePtr, ListPointer PredPtr, ListElementType
Item)
{
  ListPointer TempPtr;
  GetNode(&TempPtr,FreePtr,Node);
  if (!FullLList(TempPtr)) {
    if (PredPtr==NilValue)
    {
        Node[TempPtr].Data.AM = Item.AM;
        Node[TempPtr].Data.grade = Item.grade;
        Node[TempPtr].Next =*List;
        *List =TempPtr;
    }
    else
      {
        Node[TempPtr].Data.AM =Item.AM;
        Node[TempPtr].Data.grade=Item.grade;
        Node[TempPtr].Next =Node[PredPtr].Next;
        Node[PredPtr].Next =TempPtr;
      }
}
  else
    printf("Full List ...\n");
}

void Delete(ListPointer *List, NodeType Node[], ListPointer *FreePtr, ListPointer PredPtr)
{
  ListPointer TempPtr ;

  if (!EmptyLList(*List))
    if (PredPtr == NilValue)
    {
        TempPtr =*List;
        *List =Node[TempPtr].Next;
        ReleaseNode(Node,TempPtr,FreePtr);
    }
    else
      {
        TempPtr =Node[PredPtr].Next;
        Node[PredPtr].Next =Node[TempPtr].Next;
        ReleaseNode(Node,TempPtr,FreePtr);
```

```
197        }
198    else
199      printf("Empty List ...\n");
200  }
201
202  void TraverseLinked(ListPointer List, NodeType Node[])
203
204  {
205    ListPointer CurrPtr;
206
207    if (!EmptyLList(List))
208    {
209        CurrPtr =List;
210        while (CurrPtr != NilValue)
211        {
212            printf("[%d: (%d,%.1f) ->%d] ",CurrPtr,Node[CurrPtr].Data.AM,Node[CurrPtr].Data.grade,
Node[CurrPtr].Next);
213            CurrPtr=Node[CurrPtr].Next;
214         }
215         printf("\n");
216     }
217    else printf("Empty List ...\n");
218  }
```