```
                        }
                        break;
                case 3: Pop_temp = Pop(&s);
                        if (Pop_temp == -1)
                         printf("Stack underflow \n");
                        else
                         printf("Element Popped:%d\n", Pop_temp);
                        break;
                case 4: done = 1;
                       break;
                default:printf("Illegal Entry\n\n");
                        break;
            }
        }
}
void Push (Stack ps, int x)
{  /* Inserts an element in to the Stack */
     if  (Full(ps))  /* Check for overflow error */
     {
           printf("Error-Overflow\n");
           return;
     }
     ++(ps->top);
     ps->items[ps->top] = x; /* insert x */
}

int Pop (Stack ps)
{  /* Removes an element from the Stack */
     int temp;
     if (Empty(ps))     /* underflow */
           return(-1);
     temp = ps->items[ps->top];
     -- (ps->top);
     return(temp);
}

int Empty (Stack ps)
{  /* Returns 1 when stack is empty, else 0 */
     if (ps->top == -1)
           return 1;
     else
           return 0;
}

int Full (Stack ps)
{  /* Returns 1 when stack is full, else 0 */
     if (ps->top == MAX-1)
           return 1;
     else
           return 0;
}
```

Sample Run

```
Enter Choice: 1
Enter element: 10
        1  Insert
        2  Report
        3  Pop
        4  Exit
Enter Choice: 1
Enter element: 20
        1  Insert
        2  Report
        3  Pop
        4  Exit
Enter Choice: 3
Value of element Popped:  20
        1  Insert
        2  Report
        3  Pop
        4  Exit
Enter Choice: 2
10
```

10.3

Write a C program to convert and print a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply), and / (divide).

Program

```
#include <stdio.h>
#include <string.h>
#define MAX    80
struct stk
{
     char items[MAX];
     int top;
};
typedef struct stk * Stack;
void Push (Stack, char);
char Pop (Stack);
int  Isoperand (char);
int  f (char);
int  g (char);
int  Empty (Stack);
void Postfix (char [], char[]);
void main()
{
     char infix[MAX];
```

and the program will not search further in the table. But actually we see that element 33 is in the hash table.

To overcome this difficulty, we can employ the following methods:

- Shift all the elements when a key is deleted.
- Instead of marking the deleted location to `empty` (-1), we can use some other flag to probe the search further in the table.
- Use of another method, called as chaining.

Hashing with Chains

When a collision occurs we don't need to probe the hash table, instead of a chain or a linked list of elements are created. As in our earlier example, the hash table size M = 10 and but the elements are as shown below. With chaining, the hash table will look like as in Figure 9.5.

Elements = [20, 11, 23, 85, 83, 56, 33, 9, 16, 10, 13]

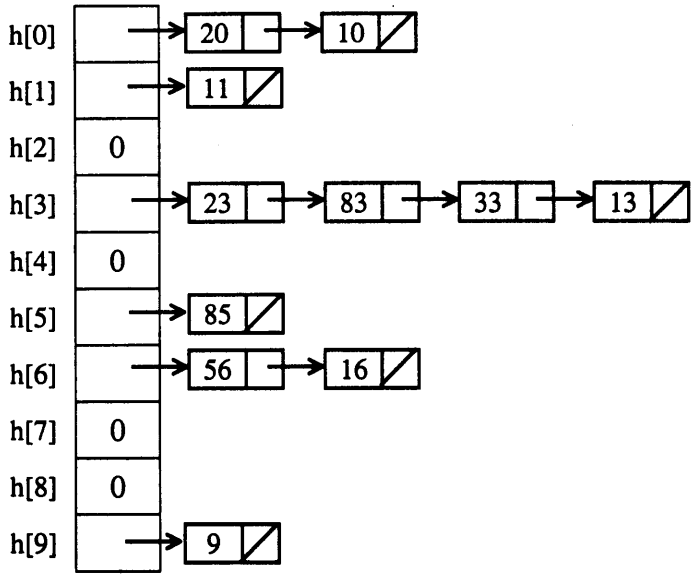

Fig. 9.5 Hashing with chains

The hash address h[3] is a linked list of 23, 83, 33 and 13. When element 23 is taken for insertion its hash address is straight away 3 and when the next element is considered, that is 83, it leads to collision and hence it is append to the linked list, and so on. The hash table is now not an integer array, but an array of linked list addresses (array of pointers). That is,

```
NODE h[M];
```

The data type NODE is defined in Section 6.6 and M is the hash table size. To put NULL in table entry 7, we use h[7] = NULL. The zeros in locations 2, 4, 7 and 8 indicate that

Write necessary functions (1) To display all the records in the file. (2) To search for a specific record based on the USN. In case the required record is not found, suitable message should be displayed. Both the options in this case must be demonstrated.

Program

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 80
struct Student {
      int USN;
      char Name[MAX];
      int Marks1;
      int Marks2;
      int Marks3;
};

void main()
{
      int i, n;
      int key, flag = 0;
      FILE *fp;
      struct Student s;

      clrscr();
      fp = fopen("std.dat", "w");
      printf ("Enter No. of student records: ");
      scanf ("%d", &n);
      printf("Enter student info:\n<USN><Name>
                  <Marks1><Marks2><Marks3>:\n");
      for (i = 0; i< n; i++)
      {
            scanf("%d%s%d%d%d", &s.USN,s.Name,
                  &s.Marks1,&s.Marks2,&s.Marks3);
            fwrite(&s, sizeof(s), 1, fp);
      }
      fclose(fp);

      fp = fopen("std.dat", "r");
      if (!fp)
      {
            printf("Error in reading file\n");
            exit(1);
      }
      /* search for student record */
      printf("Enter student USN to search:");
      scanf("%d", &key);

      while (fread(&s,sizeof(s),1,fp))
      {
            if (s.USN == key) {
```

```
                printf("%d %s %d %d %d\n", s.USN,s.Name,
                      s.Marks1,s.Marks2,s.Marks3);
                flag = 1;
            }
        }
        if (!flag) printf("Student record not found\n");
        /* display all student records */
        rewind(fp);
        printf("Student Records in the file:\n");
        printf("---------------------------\n");
        while (fread(&s,sizeof(s),1,fp))/* reads one record */
            printf("%d %s %d %d %d\n", s.USN,s.Name,
                s.Marks1,s.Marks2,s.Marks3);
        fclose(fp);
}
```

Sample Run-1

```
Enter No. of student records: 5
Enter student info:
<USN><Name><Marks1><Marks2><Marks3>:
111 Nandagopal  56 68 99
222 Deepak 87 98 77
333 Pradeep 49 100 86
444 Kavitha 0 50 3
555 Rahul 78 78 78
Enter student USN to search:333
333 Pradeep 49 100 86
Student Records in the file:
---------------------------
111 Nandagopal 56 68 99
222 Deepak 87 98 77
333 Pradeep 49 100 86
444 Kavitha 0 50 3
555 Rahul 78 78 78
```

Sample Run-2

```
Enter No. of student records: 2
Enter student info:
<USN><Name><Marks1><Marks2><Marks3>:
100 Ramprasad 89 56 78
200 Varun 90 89 62
Enter student USN to search:300
Student record not found
Student Records in the file:
---------------------------
100 Ramprasad 89 56 78
200 Varun 90 89 62
```

10.2

Write a C program to demonstrate the working of a stack of size N using an array. The elements of the stack may be assumed to be of type integer. The operations to be supported are:

(a) Push
(b) Pop
(c) Display

The program should print appropriate messages for stack overflow, stack underflow, and stack empty.

Program

```
#include <stdio.h>
#define MAX  5          /* Size of Stack  */
struct stk
{
     int items[MAX];     /* Array to hold stack elements */
     int top;            /* Stack pointer                */
};
typedef struct stk * Stack;
void Push  (Stack, int);
int  Pop   (Stack);
int  Empty (Stack);
int  Full  (Stack);

void main()
{
     struct stk s;
     int i,sp,elem,choice,done;
     int Pop_temp;
     s.top = -1;  /* initialize stack pointer */
     done = 0;
     while (!done)
     {
          printf("\t1  Insert\n\t2 Report\n\t3 Pop\n\t4  Exit\n");
          printf("Enter Choice: "); scanf("%d", &choice);
          switch (choice)
          {
               case 1: printf("Enter element: ");
                       scanf("%d", &elem);
                       Push(&s, elem);
                       break;
               case 2: if (s.top == -1)
                            printf("Empty Stack\n");
                       else   /* print stack contents */
                       {
                            for (i = s.top; i >= 0; i--)
                                 printf("%d\n", s.items[i]);
```

```
        char postr[MAX];
        printf("Enter the infix expression:\n");
        scanf("%s", infix);
        printf("The Original infix string is = %s\n", infix);
        Postfix(infix,postr);
        printf("The Postfix string is        = %s\n", postr);
}

void Postfix (char infix[], char postr[])
{
        int cur, p, len;
        int i = 0;
        char symb;
        struct stk s;

        len = strlen(infix);
        infix[len] = ')';
        infix[++len] = '\0';
        s.top = 0;
        s.items[s.top] = '(';
        for (cur = 0; (symb = infix[cur]) != '\0'; cur++)
        {
                while (f(symb) < g(s.items[s.top]))
                        postr[i++] = Pop(&s);
                if (f(symb) != g(s.items[s.top]))
                        Push(&s, symb);
                else
                        Pop(&s);
        }
        postr[i] = '\0';
}

int Isoperand (char symb)
{
        if ((symb >= 'a') && (symb <= 'z'))
                return(1);
        else
                return(0);
}

int f (char op1)
{
        if (Isoperand(op1)) op1 = '#';
        switch (op1)
        {
                case '+':
                case '-': return(1);
                case '*':
                case '/': return(3);
                case '$': return(6);
                case '#': return(7);
                case '(': return(9);
                case ')': return(0);
```

```
        struct stk opndstack;
        opndstack.top = -1;
        for (pos = 0; (c = expr[pos]) != '\0'; pos++)
              if (isdigit(c)) /* return true if c is 0 to 9 */
                    Push (&opndstack, (float) (c - '0'));
              else
              {
                    opnd2 = Pop(&opndstack);
                    opnd1 = Pop(&opndstack);
                    value = oper(c,opnd1,opnd2);
                    Push(&opndstack,value);
              }
              return(Pop(&opndstack));
}

float oper (char symb, float op1, float op2)
{  /* Returns float after appropriate operation */
      switch (symb)
      {
              case '+' : return (op1 + op2);
              case '-' : return (op1 - op2);
              case '*' : return (op1 * op2);
              case '/' : return (op1 / op2);
              case '$' : return (pow(op1, op2));
              default  : printf("%s","Illegal operator");
                         exit(1);
      }
}

void Push (Stack ps, float  x)
{
      ++(ps->top);
      ps->items[ps->top] = x;
}

float Pop (Stack ps)
{
      float temp;
      if (ps->top == -1) return (-1);
      temp = ps->items[ps->top];
      --(ps->top);
      return (temp);
}
```

Sample Run

```
the postfix expression:
253/+
The original postfix expr. is = 253/+
3.666667
```

10.5

Write a C program to simulate the working of a queue of integers using an array. Provide the following operations:

(a) Insert
(b) Delete
(c) Display

Program

```
#include <stdio.h>
#define MAX   5 /* maximum size of Queue */
struct que
{
     int items[MAX];
     int front;
     int rear;
};
typedef struct que * Queue;
void Qinsert (Queue, int);
int  Qdelete (Queue);
int  QEmpty (Queue);
int  QFull (Queue);

void main()
{
     struct que q;
     int i,elem,choice,done;
     int Qdelete_temp;
     q.front = q.rear = -1; /* initailize */
     done = 0;
     while (!done)
     {
          printf("\t1 Insert\n\t2  Report\n\t3
                    Delete\n\t4  Exit\n");
          printf("Enter Choice: ");
          scanf("%d", &choice);
          switch (choice)
          {
               case 1: printf("Enter element: ");
                       scanf("%d", &elem);
                       Qinsert(&q, elem);
                       break;
               case 2: if (q.front == -1)
                       {
                            printf("Empty Queue\n");
                            break;
                       }
                       else
                       {
```

```
        int front;
        int rear;
        int items[MAX];
};
typedef struct cir_queue * CQueue;
void Initialize (CQueue);
void CQinsert (CQueue, int);
int  CQdelete (CQueue );
void Display (CQueue);
int  CQEmpty (CQueue);
int  CQFull (CQueue);

void main()
{
        struct cir_queue cq;
        int i,sp,elem,choice,done;
        int CQdelete_temp;
        /* initailize count, front and rear pointers */
        Initialize(&cq);
        done = 0;
        while (!done)
        {
         printf("\t1  Insert\n\t2  Report\n\t3  Delete\n\t4  Exit\n");
         printf("Enter Choice: ");
         scanf("%d", &choice);
         switch (choice)
         {
              case 1: printf("Enter element: ");
                      scanf("%d", &elem);
                      CQinsert(&cq, elem);
                      break;
              case 2: Display(&cq);
                      break;
              case 3: CQdelete_temp = CQdelete(&cq);
                      if (CQdelete_temp == -1)
                          printf("CQueue Underflow\n");
                      else
                          printf("Deleted element:
                                 %d\n",CQdelete_temp);
                          break;
               case 4: done = 1;
                       break;
               default:printf("Illegal Entry\n\n");
                       break;
         }
       }
}
void Initialize (CQueue cq)
{  /* Initializes the cir_queue members */
        cq->count = 0;
        cq->front = 0;
        cq->rear  = -1;
}
```

```
void CQinsert (CQueue cq , int x)
{  /* Inserts x in the Queue */
     if (CQFull(cq))   /* check for overflow */
     {
           printf("CQueue Full\n");
           return;
     }
     cq->count++;     /* update count */
     cq->rear = (cq->rear + 1) % MAX;
     cq->items[cq->rear] = x;
}

int CQdelete (CQueue cq )
{  /* Deletes an element from Queue */
     int temp;

     if (CQEmpty(cq))
           return(-1);   /* underflow    */
     cq->count--;        /* update count */
     temp = cq->items[cq->front];
     cq->front = (cq->front + 1) % MAX;
     return(temp);
}

void Display (CQueue cq)
{
     int i, f;

     if (!CQEmpty(cq))
     {
           f = cq->front;
           for (i = 1; i <= cq->count; i++)
           {
                 printf("%d ", cq->items[f]);
                 f = (f + 1) % MAX;
           }
           printf("\n");
     }
     else  printf("CQueue Empty\n");
}

int CQFull (CQueue cq)
{
     if (cq->count == MAX) return 1;
     else  return 0;
}

int CQEmpty (CQueue cq)
{
     if (cq->count == 0)  return 1;
     else  return 0;
}
```

```
            printf("Queue %d - Overflow\n",p);
            return;
      }
      q[p].items[++q[p].rear] = x;
      if (q[p].front == -1)
            q[p].front = 0;
}

/* Deletes an element from Queue */
int Qdelete (Queue q[])
{
      int temp = -1;
      int empty, p;
      for (p = 0; p < n; p++)
      {
            empty = QEmpty(q,p);
            if (!empty)
            {
                  temp = q[p].items[q[p].front];
                  if (q[p].front == q[p].rear)
                        /* reset the pointers */
                        q[p].front = q[p].rear = -1;
                  else
                        q[p].front++;
                  break;
            }
      }
      return temp;
}

int QFull (Queue q[], int p )
{
      if (q[p].rear == MAX - 1) return 1;
      else  return 0;
}

int QEmpty (Queue q[], int p)
{
      if (q[p].front == -1) return 1;
      else  return 0;
}
```

Sample Run

```
Enter the priority & the element: 1 10
        1  Insert
        2  Report
        3  Delete
        4  Exit
Enter Choice: 1
Enter the priority & the element: 2 21
        1  Insert
        2  Report
```

```
        3  Delete
        4  Exit
Enter Choice: 2
Queue : 0
10
Queue : 1
20 21
Queue : 2
Enter Choice: 3
Deleted element: 10
        1  Insert
        2  Report
        3  Delete
        4  Exit
Enter Choice: 2
Queue : 0

Queue : 1
20 21
Queue : 2
```

10.8

Write a C program using dynamic variables and pointers to construct a singly linked list consisting of the following information in each node: student id (integer), student name (character string), and semester (integer). The operations to be supported are:

(a) The insertion operation.
- **(i) At the front of the list.**
- **(ii) At the back of the list.**
- **(iii) At any position in the list.**

(b) Deleting a node based on student id. If the specified node is not present in the list an error message should be displayed. Both the options must be demonstrated.

(c) Searching a node based on student id and update the information content. If the specified node is not present in the list an error message should be displayed. Both the options must be demonstrated.

(d) Displaying all nodes in the list.

(Note: The question may be asked as one of a/b/c with d).

Program

```
#include <stdio.h>
#include <alloc.h>
#include <string.h>
#define MAX 30
```

```
      q->link = 0;
      if (!first) first = q;
      else {
            while (t->link)
                  t = t->link;
            t->link = q; /* goto last node */
           }
      return first;
}

NODE InsPos (NODE first)
{
      NODE q, t = first;
      int k; /* key */
      int sid;
      char sname[MAX];
      int ss;
      printf("Enter key Std id:");
      scanf("%d", &k);
      while (t && t->Std_id != k) t = t->link;
      if (t)
      {
            printf("Enter Student id: ");
            scanf("%d", &sid);
            fflush(stdin);
            printf("Enter Student Name: ");
            gets(sname);
            fflush(stdin);
            printf("Enter Semester: ");
            scanf("%d", &ss);
            q = (NODE) malloc (sizeof(struct List));
            q->Std_id = sid;
            strcpy (q->Std_name, sname);
            q->Sem = ss;
            q->link = t->link;
            t->link = q;
      }
      else printf ("Std id not found - insertion not possible\n");
      return first;
}

/* To delete a job from the list, given its id number */
NODE DelStd (NODE first , int x)
{
      NODE current, pred;

      /* single node case */
      if (first->Std_id == x)
      {
            first = first->link;
            return first;
      }
```

```
        /* more than one node case */
        pred = first;
        current = first->link;
        while (current && current->Std_id != x)
        {
              current = current->link;
              pred = pred->link;
        }

        if (current)
              pred->link = current->link; /* remove the node */
        else printf(" Student id not Found \n");
        return first;
}

/* Displays the contents of the list */
void Display (NODE first)
{
        while (first)
        {
              printf("%d\t%s\t%d\n", first->Std_id, first->Std_name,
                                    first->Sem);
              first = first->link;
        }
}

/* To search for a Student */
int Search (NODE q, int key)
{
        char newn[MAX];
        int  news;

        while (q && q->Std_id != key) q = q->link;
        if (q)    /* Std id found     */
        {
              printf("Enter name to update:");
              scanf("%s", newn);
              strcpy(q->Std_name, newn);
              fflush(stdin);

              printf("Enter sem to update:");
              scanf("%d", &news);
              q->Sem = news;
              return 1;
        }
        else return 0;          /* Std id not found */
}
```

Sample Run

```
Enter Student id: 201
Enter Student Name: Divya
Enter Semester: 3
```

```
                    case 4: done=1;
                            break;

                    default:printf("**** Illegal Entry ***\n\n");
                            break;
                }
        }
}

NODE Push (NODE first, int x)
{
      NODE q;
      q = (NODE) malloc (sizeof(struct List));
      q->info = x;
      q->link = first;
      first = q;
      return first;
}

NODE Pop (NODE first , int *x, int *flag)
{
      *flag = 1;
      if (first)
      {
            *x = first->info;
            first = first->link;
      }
      else *flag = 0;
      return first;
}

/* Display the ist */
void Display (NODE first)
{
      while (first)
      {
            printf("%d\n", first->info);
            first = first->link;
      }
}
```

Sample Run

```
1       Push
2       Report
3       Pop
4       Exit
Enter Choice: 1
Enter info: 11

1       Push
```

```
2       Report
3       Pop
4       Exit
Enter Choice: 1
Enter info: 22
Enter Choice: 1
Enter info: 33

1       Push
2       Report
3       Pop
4       Exit
Enter Choice: 3
Popped element is: 33

1       Push
2       Report
3       Pop
4       Exit
Enter Choice: 3
Popped element is: 22
Enter Choice: 2
11

1       Push
2       Report
3       Pop
4       Exit
Enter Choice: 3
Popped element is: 11

1       Push
2       Report
3       Pop
4       Exit
Enter Choice: 3
Stack empty
```

10.10

Write a C program using dynamic variables and pointers to construct a queue of integers using singly linked list and to perform the following operations:

(a) Push
(b) Pop
(c) Display

Program

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

```
1       QInsert
2       Report
3       QDelete
4       Exit
Enter Choice: 1
Enter info: 30
10
20
30
1       QInsert
2       Report
3       QDelete
4       Exit
Enter Choice: 3
Deleted element is: 10
1       QInsert
2       Report
3       QDelete
4       Exit
Enter Choice: 3
Deleted element is: 20
1       QInsert
2       Report
3       QDelete
4       Exit
Enter Choice: 3
Deleted element is: 30
1       QInsert
2       Report
3       QDelete
4       Exit
Enter Choice: 3
Queue empty
```

10.11

Write a C program to support the following operations on a doubly linked list where each node consists of integers.

(a) Create a doubly linked list by adding each node at the front.

(b) Insert a new node to the left of the node whose key value is read as an input.

(c) Delete the node of a given data, it is found, otherwise display appropriate message.

(d) Display the contents of the list.

Program

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