4.3

StrLength(s) = 14

StrLength(t) = 4

SubString(s, 8, 7) = ‘STUDENT’

SubString(t, 2, 1) = ‘O’

Index(s, ‘A’) = 3

Index(s, t) = 0

Replace(s, ‘STUDENT’, q) = ‘I AM A WORKER’

Concat(SubString(s, 6, 2), Concat(t, SubString(s, 7, 8))) = ‘A GOOD STUDENT’

4.4

s = ‘THIS SAMPLE IS’

t = ‘A GOOD’

v = ‘THIS A GOOD ONE’

StrLength(s) = 14

Index(v, g) = 3

Index(u, g) = 0

4.8

#include<stdio.h>

#include<stdlib.h>

#define MAXSTR 100

typedef struct{

char str[MAXSTR];

int length;

}StringType;

void ClearString(StringType s);

void StrAssign(StringType t, StringType s);

int StrCompare(StringType s, StringType t);

int StrLen(StringType s);

StringType Concat(StringType s, StringType t);

StringType SubString(StringType s, int start, int len);

void ClearString(StringType s)

{

s.length = 0;

s.str[0] = '\0';

}

void StrAssign(StringType t, StringType s)

{

ClearString(t);

t.length = s.length;

int i;

for(i = 0; i < s.length; i++)

t.str[i] = s.str[i];

}

int StrCompare(StringType s, StringType t)

{

int i = 0;

int value;

for(i = 0; i < s.length && i < t.length; i++)

{

value = s.str[i] - t.str[i];

return (value > 0) ? 1: -1;

}

value = s.length - t.length;

if(value == 0)

return 0;

else

return (value > 0) ? 1: -1;

}

int StrLen(StringType s)

{

return s.length;

}

StringType Concat(StringType s, StringType t)

{

StringType p;

p.length = s.length + t.length;

int i;

for(i = 0; i < p.length; i++)

{

if(i < s.length)

p.str[i] = s.str[i];

else

p.str[i] = t.str[i - s.length];

}

return p;

}

StringType SubString(StringType s, int start, int len)

{

if(1 > start || start > s.length || len < 0 || len > s.length - start + 1)

exit -1;

StringType t;

int i;

for(i = 0; i < len; i++)

t.str[i] = s.str[start + i - 1];

t.length = len;

return t;

}

4.10

StringType reverse(StringType s){

int i, j;

StringType r;

for(i = StrLen(s); i; i--)

{

r.str[StrLen(s) - i] = s.str[i - 1];

}

return r;

}

4.11

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAXSTR 100

typedef struct{

char str[MAXSTR];

int length;

}StringType;

typedef struct{

char str[MAXSTR];

int pos[MAXSTR];

int length;

}StringType2;

StringType2 pick(StringType s, StringType t);

int main()

{

StringType s={{"abcddee"},7};

StringType t={{"bsd"},3};

StringType2 r;

int i;

r = pick(s, t);

for(i = 0; i < r.length; i++)

printf("%c",r.str[i]);

printf("\n");

for(i = 0; i < r.length; i++)

printf("%d ", r.pos[i]);

printf("\n");

return 0;

}

StringType2 pick(StringType s, StringType t)

{

int i, j, k = 0;

char c;

StringType2 r;

for(i = 0; i < s.length; i++)

{

c = s.str[i];

for(j = 0; j < i && c != s.str[j]; j++);

if(i == j)

{

//printf("%c\n", c);

for(j = 0; j < t.length && t.str[j] != c; j++);

//printf("%d\n", j);

if(j == t.length)

{

r.str[k] = s.str[i];

r.pos[k] = i + 1;

k++;

}

}

}

r.length = k;

return r;

}

4.16

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAXSTR 100

typedef struct{

char str[MAXSTR];

int length;

}StringType;

int StrCompare(StringType s, StringType t)

{

int i = 0;

int value;

for(i = 0; i < s.length && i < t.length; i++)

{

value = s.str[i] - t.str[i];

return (value > 0) ? 1: -1;

}

value = s.length - t.length;

if(value == 0)

return 0;

else

return (value > 0) ? 1: -1;

}

4.17

StringType Replace(StringType s, StringType t, StringType v)

{

int i, j, k;

for(i = 0; i < s.length; i++)

{

for(j = i, k = 0; k < t.length && t.str[k] == SubString(s,i,t.length).str[k];k++,j++);

if(k == t.length)

{

if(t.length == v.length)

for(j = 0; j < t.length; j++)

s.str[j + i - 1] = v.str[j];

else if(t.length > v.length)

{

for(j = 0; j < s.length-i-t.length+1; j++)

s.str[i+j+v.length-1] = s.str[i+j+t.length-1];

s.str[s.length-t.length+v.length]='\0';

for(j = 0; j < v.length; j++)

s.str[i+j-1] = v.str[j];

}

else

{

for(j = s.length+v.length-t.length; j > i+v.length-2; j--)

{

s.str[j] = s.str[j + t.length - v.length];

}

s.length += v.length -t.length;

for(j = 0; j < v.length; j++)

s.str[i+j-1] = v.str[j];

}

}

}

return s;

}

4.23

#include<stdio.h>

#include<stdlib.h>

#define CHUNK\_SIZE 4

typedef struct Chunk{

char ch[CHUNK\_SIZE];

struct Chunk \* next;

}Chunk;

typedef struct{

Chunk \*head, \*tail;

int curlen;

}LString;

int Palindrome(LString \*s)

{

Chunk \*p;

p = s->head;

Stack l;

char c;

int i = 0, k = 0;

for(k = 0; k < s->curlen; k++)

{

if(k <= s->curlen/2)

push(&l, p->ch[i]);

else if(k > (s->curlen)/2)

{

pop(&l, &c);

if(p->ch[i] != c)

return 0;

}

if(++i == CHUNK\_SIZE)

{

p = p->next;

i = 0;

}

}

return 1;

}

4.29

LStrNode \* KMP2(LString t, LStrNode \*pos)

{

LStr \*p, \*q;

int i;

p = pod;

q = t->succ;

while(p && q)

{

if(q == t || p->chdata == q->chdata)

{

p = p->next;

q = q->next;

}

else

q = q->next;

}

if(!q)

{

for(i = 0; i < StrLen(t); i++)

p = p->next;

return p;

}

return NULL;

}

4.30