一.选择题

CADACDDACB

二. 简答题

1.答：（教材 P92)采用顺序存储结构，在重新调整存储空间时，元素的移动 S 比较大，尤其是在分配的存储空间即将充满时．而采用链式存储结构就可以避免这种情况．另外，由于采用了链式存储结构，不必事先声明一片存储区作为堆栈的存储空间，因而不存在因为栈满而产生溢出的问题，只要从存储库中可以动态申请到可用链结点空间，就可以认为堆栈没有满．在实际问题中，若不知道或难以估计将要进栈元素的最大数量时，采用链式存储结构比采用顺序存储结构更为合适。

2.答：（教材 P170)根据定义，二叉树的前序遍历是先访问根结点，其次再按前序遍历方式递归地遍历根结点的左子树和右子树．显然，在得到的前序序列中，第一个结点一定是二叉树的根结点．根据二叉排序树的特点，每个结点的值都大于它的左子树所有结点的值(若存在的话），都小于它的右子树所有结点的值（若存在的话）．二叉排序树的这一特点自然而然地将前序序列分割成前后两个子序列，即左右两个子树．如此递归下去，当取尽前序序列中的所有结点时，便得到了一棵二叉排序树。

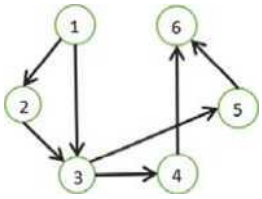
3.答：（教材 P288)至少要进行 0+1+.....+(n-1)=n(n-1)/2 次探测．因为，散列表初始为空，第一次向散列表插入关键字时无冲突，无需进行探测：第二次时有冲突，需要探测一次;第三次时有冲突，需要探测两次．以此类推．因此至少要进行 n(n-1)/2 次探测。

4.答：（教材 P319)不一定是，因为后续的排序过程中可能还有更小的元素会插入到它的前面。

三. 综合题

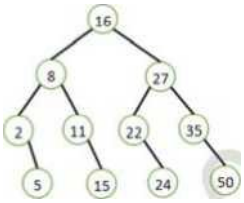
1. 这段代码的功能是删除链表中值域为 x 的结点。

2. 如图，



拓扑序列有：1,2,3,4,5,6; 1,2,3,5,4,6

3.判定树如图所示，



比较一次的分别有：16;

比较两次的分别有：8,27;

比较三次的分别有：2,11,22,35;

比较四次的分别有：5,15,24,50

4.该序列为：(65,60,63,50,36,40,25,38)

四．算法设计题

```
#define M 50
void ExchangeChild(BTREE T)
{
        BTREE STACK[M],p=T,q=NULL;
```

```
        int top=-1;
        if(T!=NULL)

        {
            STACK[++top]=p;
            while(!(p==NULL&&top==-1))
            {
                p=STACK[top--];
                if(p!=NULL)
                {
                    if(p->data==item)
                    {
                        q=p->lchild;
                        p->lchild=p->rchild;
                        p->rchild=q;
                        break;
                    }
                    STACK[++top]=p->rchild;
                    STACK[++top]=p->lchiId;
                }
        }
    }
}
```

五．单项选择题

DBCADADCCB

六．综合题

1. a=(x-x%100)/100;b=(x-x%10-a*100)/10;c=x-100*a-10*b;
2. (1) s[n】==’ls【n]=’\t’ls[n】==’\n’(2)n=n-1(3)n
3. 静态局部变量采用静态存储方式，在程序运行期间由系统分配固定的存储空间．普通局 部变量采用动态存储方式，在程序运行期间根据需要进行动态地分配存储空间．如果在定义 局部变量时不赋初值的话，对于静态局部变量来说，编译时自动赋初值 0 或空字符．而对普 通局部变量来说，它的值是一个不确定的值．
4. 预处理指令是由 ANSI C 统一规定的用以改进程序设计环境，提高编程效率的指令．它 不是 C 语言本身的组成部分，不能直接对它们进行编译．必须在对程序进行编译之前根据预 处理指令对程序作相应的处理．预处理指令的典型应用场景有宏定义和文件包含．

七．程序设计题

```
int get_weekday(int year,int month,int day)
{
    int y=year,m=month,d=day;
    int leaps=0,year_leap=0,days=0;
    int year_isleap[12]={31,29,31,30,31,30,31,31,30,31,30,31};
    int year_isnotleap[12】={31,28,31,30,31,30,31,31,30,31,30,31};
```

```
        for( ;y>=1;y--)
        {
                if(y%4==0)
                {
                        if(y%100==0)
                        {
                                if(y%400==0)
                                {
                                        if(y==year)
                                                year_leap-1;
                                        else
                                                leaps++;
                                }
                        }
                        else
                        {
                                if(y==year)
                                        year_leap=1;
                                else
                                        leaps++;
    }
     }
}
```

```
    days+=leaps*366+(year-1-leaps)*365;
    if(year_leap)
            for(m=m-2;m>=0;m--)
                    days+= year_isleap[m];
    else
           for(m=m-2;m>=0;m--)
                  days+= year_isleap[m];
    days+=d;


     return (days%7);
}

int a=get_weekday (1,1,5),          int a=get_weekday(1,2,5);
                                                Printf(" &d\n" ,a);
```

八. 程序设计题

```
#includc <stdio.h>
#include <stdlib.h>
#include <string.h>
void main()
{
    char ltem[20],
    filename[20],
    ch[20],/*存放待比较的单词*/
    com_item[20],/*用于比较过程中做标记*/
    output_ch[20];/*已经输出过的单词*/
    FILE*fp;/*指向进行对比的文件*/
    FILE*output_fp=fopen(“output”,“w+”);/*指向暂时存储已经输出的字符串的文件*/
int i=0,j=0,is_output=0,is_finded=0;
printf(“请输入您要查找的单词：\n”);
scanf(“%s”,item);
for(i=0;item[i]!='\0';i++)/*转为小写*/
if(item[i]>=65&&item[i]<=90)
item[i]=item[i]+32;
printf(“请输入目标文件：\n”);
scanf(“%s”,filename);
if((fp=fopen(filename,“r”))==NULL)
{
    printf(“cannot open file”);
}
fscanf(fp,“%s”,ch);
while (! feof (fp) )
{
```

```
for(i=0;ch[i】!='\0';i++)/* 转为小写*/
        if(ch[i]>=65&&ch[i]<=90)
                ch[i]=ch[i]+32;
if((strlen(item)!=strlen(ch))| |strcmp(item,ch)==0)
{
        fscanf(fp,"%s",ch);
        continue;/*若长度不相等或为原字符串，则跳过，节省时间*/

}

for(i=0;item[i] !='\0';i++)
        com_item[i]=item[i];/*复制到另一个数组中，便于做标记*/
for(i=0;ch[i] !='\0';i++)
```

```
{

        for(j=0;com_item[j】！ =ch[i】&&com_item[j】！ =’\0’ ;j==_;
        if(com_item[j]==’\0’
                break;/*若未找到对应字母，则跳出本次检查*/
        else
                com_item[j]='*';/*若找到对应字母，则做标记*/
 }
 if(i==strlen(ch))/*若该单词是变位单词，则判断是否输出*/
 {
          is_finded=1;
          fseek(output_fp,0,0);/*从头开始搜索存放已经输出的单词的文档*/
          fscanf(output_fp,"%s",output_ch);
          while( !feof(output _fp))
           {

                if(strcmp(output_ch,ch)==0)

      {

       is_output=1;

                        break;/*若已经输出过相同的单词，则标记为 1，且不

                      再继续查找*/

}

              fscanf(output_fp,"%s",output_ch);
           }
          if(!is_output)
           {
                    fprintf(output_fp,"% s\n",ch);
                    printf("%s\n",ch);
           }
          is _output=0;

 }

fscanf(fp,"%s",ch);

}

if(is_finded==0)
```

```
                printf("Not find the right word"); fclose(fp);
    fclose(output_fp);
}
```