浙江大学 2013－2014 学年夏季学期

《C 程序设计专题》课程期末考试试卷

课程号：__211Z0050___， 开课学院：__计算机学院__

考试试卷：√A 卷、B 卷（请在选定项上打√）

考试形式：√闭、开卷（请在选定项上打√），允许带__/__入场

考试日期：__2014__年__06__月__27__日，考试时间：__120__分钟

诚信考试，沉着应考，杜绝违纪.

考生姓名：_____学号：_____所属院系：_____

(注意：答题内容必须写在答题卷上，写在本试题卷上无效)

Section 1: single Choice(2 marks for each item, total 26 marks)

1. Suppose that five characters are pushed into a stack in the order of ***a, b, c, d, e***. The impossible popping sequence is ___.
   A. a b c d e B. e d c b a C. d c e a b D. d e c b a
2. When using a stack to evaluate the postfix expression ***2 3 5 + 6 * + 8 4 / +***, what symbols are inside the stack when "/" is read?___.
   A. +*+ B. ++ C. 2 3 5 6 8 4 D. 50 8 4
3. To insert a node pointed by ***s*** after the node pointed by ***p*** in a linked list, we should do the following statements ___.
   A. s->next=p; p->next=s; B. s->next=p->next; p->next=s;
   C. s->next=p->next; p=s; D. p->next=s; s->next=p;
4. Assume that two structures are defined as following:

```
struct data {
    int day, month, year;
};
struct student {
    char name[20];
    long num;
    struct data birthday;
};
```

   And ten such structures are allocated:
   struct student *p = (struct student *)malloc(10 * sizeof(struct student));
   The expression ___ below can set the year of the second student's birthday to 1995 correctly.
   A. (p+1)->birthday.year = 1995; B. *(p+1)->birthday.year = 1995;
   C. (p+1)->birthday->year = 1995; D. (p+1).birthday->year = 1995;
5. About the function pointer, the statement ___ below is WRONG.
   A. The function pointer can be passed to a function as an actual argument.
   B. The function pointer can be used to call a function.
   C. Can't pass argument to a function when a function pointer is used to call the function.
   D. The function name is a pointer value which can be assigned to a function pointer variable.

6. About ***#include***, the statement___ below is correct.
   A. #include "xx.h" means that search the file xx.h in the current directory ONLY.
   B. #include <stdio.h> means that the ***stdio*** library is included in the program.
   C. #include <xx.h>means that search the file xx.h ONLY in the directory which be appointed by the compiler.
   D. #include<xx> is NOT correct because of missing the postfix ***.h***
7. Given the variables declaration ***char s[5]***;.If we convert the integer variable ***i*** to the corresponding character string, the statement ____ below is correct.
   A. sprintf(s, "%d", i); B. sprintf(s, "%5d", i);
   C. sprintf(s, "%4d", i); D. None of the above.
8. For the graphics library of our course, the statement ___ about the function ***printf*** below is correct.
   A. Printf can NOT output anything.
   B. Printf can output the content in the current position of the graphics window.
   C. Printf can open a text window and output the content automatically.
   D. If calling the function InitConsole(), printf will output the content in the special console window, otherwise nothing will be output.
9. Use ***typedef*** to define a function pointer type **PunPtr** which points to a function ***void func(int x)***. The correct expression is___.
   A. typedef FunPtr fun(int x); B. typedef void FunPtr(int x);
   C. typedef void (*FunPtr)( ); D. typedef void (*FunPtr)(int);
10. The complexity(复杂度) of sort algorithm ___ below is $O(N*Log_2N)$.
    A. Selection sort algorithm B. Bubble sort algorithm
    C. Insertion sort algorithm D. Merge sort algorithm
11. In the selection sort function ***SelectionSort(int a[],int n)***, a function ***swap*** is used to exchange two variables' values. The correct swap function which can fulfill(实现) the function(功能) is___.

```
A. void swap(int a, int b) {
       int t;
       t=a;a=b;b=t;
   }
B. void swap(int *a, int *b) {
       int *t;
       t=a;a=b;b=t;
   }
C. void swap(int a[], int i, int j) {
       int t;
       t=a[i];a[i]=a[j];a[j]=t;
   }
D. void swap(int *a, int *b) {
       int *t;
       *t=*a;*a=*b;*b=*t;
   }
```

12. About the recursive function, the statement ___ below is WRONG.
    A. The recursive function must call the function itself DIRECTLY.
    B. The recursion must have an EXIT(出口).
    C. The sub-problems in a recursive solution have the same form as the original problem.
    D. The C systems use the STACK strategy to implement the recursion .
13. Given the following definitions, the ***scanf*** statement _____ below is NOT correct.

```
struct pupil {
    char name[20];
    int age;
    int sex;
} pup[5], *p=pup;
```

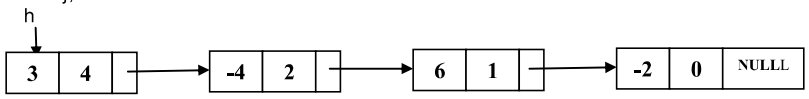
    A. scanf("%s",pup[0].name); B. scanf("%d",&pup[0].age);
    C. scanf("%d", p->age); D. scanf("%d",&(p->sex));

Section 2: Read the following problems and answer questions (5 marks for each item, total 30 marks)

1. Given the polynomial(多项式) definition of a linked list and a practical example (linked list ***h***), and the function ***f*** which is a mathematical operation(数学运算) on the

polynomial. Please draw up(画出) the result (a new polynomial) of the linked list ***h*** after calling the function ***f(h)***.

```
struct ListNode {
    int fact; /*coefficient(系数)*/
    int exp; /*exponent(指数)*/
    struct ListNode *next;
};
```




```
void f(struct ListNode *head)
{
    struct ListNode HEAD;
    struct ListNode *cur = head,  *pre = &HEAD;

    HEAD.next = head;
    If (!head) return;
    while(cur) {
        if(cur->exp) {
            cur->fact = cur->fact * cur->exp;
            cur->exp = cur->exp - 1;
        } else {
            free(cur);
            pre->next = NULL;
            break;
        }
        pre = cur;
        cur = cur->next;
    }
}
```

2. Given the definition of a linked list and a practical example (list ***h***), please write out the result of the linked list after calling the function ***f(h,3)***.

```
struct ListNode {
    int val;
    struct ListNode *next;
};
struct ListNode *f(struct ListNode *head, int x)
{
    struct ListNode *root = (struct ListNode *)malloc(sizeof(struct ListNode));
    struct ListNode *pivot = (struct ListNode *)malloc(sizeof(struct ListNode));
    struct ListNode *root_last = root;
    struct ListNode *pivot_last = pivot;
    struct ListNode *current = head;

    memset(root, 0, sizeof(struct ListNode));   /*Initial root spaces as zeroes*/
    memset(pivot, 0, sizeof(struct ListNode)); /* Initial pivot spaces as zeroes */
    while (current != NULL) {
        if (current->val < x) {
            root_last->next = current;
            root_last = current;
        } else {
            pivot_last->next = current;
            pivot_last = current;
        }
        current = current->next;
    }
    root_last->next = pivot->next;
    pivot_last->next = NULL;
    return root->next;
}
```

The input linked list ***h*** is:
1->4->3->2->5->2 and x = 3.

3. The following program will output_____.

```
#include <stdio.h>
int f(int x, int n, char s[])
{
   int count;

   if (x < 9) {
      s[n] = x + '0';
      s[n+1] = '\0';
      return n+1;
   }
   count = f(x/9, n, s);
   count = f(x%9, count, s);
   return count;
}
main()
{
    int a = 159;
    char s[100];

    f(a, 0, s);
    puts(s);
}
```

4. Write down the declaration of the function pointer variable ***p*** which points to a procedure(过程) ***f*** with two formal parameters: the first is an integer variable, the other is an integer pointer array.

5. Xiao Ming wrote a C program to call the library function ***printf("hello\n")*** in the main function , but he forgot to write down ***#include <stdio.h>*** in the beginning of the source file. However his program was be compiled and executed correctly. Why? Give a more example to explain that in what situation the program can be compiled but can NOT be executed correctly if the related head files are not be included.

6. Please describe ***the linear search algorithm***(线性查找算法) and ***the binary search algorithm***(二分查找算法). Evaluate their efficiencies(效率) and explain their use conditions(使用条件) respectively.

Section 3: According to the specification, complete each program (2 marks for each blank, total 24 marks)

1. Given a sorted linked list in ascending order（升序）, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list.

   For example,

   Given 1->2->3->3->4->4->5, return 1->2->5.

Given 1->1->1->2->3, return 2->3.

```
typedef struct _ListNode {
      int val;
      struct _ListNode *next;
} ListNode;

ListNode *insert(ListNode *target, ListNode *head, ListNode *tail)
   /*insert a node target after the tail of list which has a head node */
{
    if(!head->next)  /*Null list – only head node.*/
       head->next = target;   /*insert after head, as the first node */
    else
       tail->next = target;   /*insert after the tail */
    tail = target;
    tail->next = ___(1)___;
    return tail;
}

ListNode *deleteDuplicates(ListNode *head)
   /* head is the sorted linked list which has no head node */
{
      ListNode result_head, *result_tail, *pre = head, *cur;

      if(!head || !head->next) return head;
      result_head.next = NULL;
      cur = head->next;
      while(pre) {
           while (cur && cur->val == pre->val)
                cur = cur->next;
           if (cur ==___(2)___)
                result_tail = insert(pre, &result_head, result_tail);
           pre = cur;
           if (cur)  cur = cur->next;
      }
      return ___(3)___;
}
```

2. Merge two sorted linked lists and return it as a new sorted list. The new list should be made by splicing(拼接) together the nodes of the first two lists. The sorted list is in ascending order.

```
typedef struct _ListNode {
      int val;
      struct _ListNode *next;
} ListNode;

ListNode *mergeTwoLists(ListNode *l1, ListNode *l2)
{
    ListNode HEAD, *p;
    ListNode *cur = &HEAD;

    while (l1 && l2) {
        if(___(4)___) {
            cur->next = l1;
            l1= l1->next;
        } else {
```



```
            cur->next = l2;
            l2 =___(5)___;
        }
        cur = cur->next;
    }
    p =___(6)___;
    cur->next = p;
    return HEAD.next;
}
```

3. The graphics library of our course uses a timer(定时器) strategy which contains three functions：
   - void RegisterTimerEvent(TimerEventCallback callback); /*register the callback function*/
   - void starttimer(int timerID, int timeinterval); /*start the timer*/
   - void canceltimer(int timerID); /*close the timer*/

Now we wants to implement a single timer task(单次定时任务) that outputs a string ***"Hello World"*** in the console window after 1000 ms, then the display will not occur again. Please complete the following program fragment.
NOTE: If you think that some places don't need to be filled in anything, you can leave them blanks. Don't care about the codes which create the window.

```
void timerCallBack(int timeID)
{
      printf("Hello World\n");
      ___(7)___;
}
……
void Main()
{
      .....
      RegisterTimerEvent(___(8)___);
      starttimer(___(9)___);
      .....
}
```

4. The following program uses the recursive quick sort algorithm to sort the character strings in ascending order（升序）according to the compare strategy in the function ***numcmp***. The program will output:

```
3
22
111
```

Complete the program.

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

/* numcmp: compare s1 and s2 numerically */
int numcmp(char *s1, char *s2)
{
    int v1, v2;

    v1 = atoi(s1);
    v2 = atoi(s2);
    if (v1 < v2)
        return -1;
    else if (v1 > v2)
```

```
        return 1;
    else
        return 0;
}

void swap(char *v[], int i, int j)
{
    char *temp;

    temp = v[i];
    v[i] = v[j];
    v[j] = temp;
}

/* qsort: sort v[left]...v[right] into increasing order */
void qsort(char *v[], int left, int right, int (*comp)(char *, char *))
{
    int i, last;
    void swap(char *v[], int, int);

    if (left >= right)        /* do nothing if array contains nothing*/
        return;
    swap(v, left, (left + right)/2);
    last = left;
    for (i = left+1; i <= right; i++)
        if((*comp)(v[i], v[left]) < 0)
            swap(v, ++last, i);
    swap(v, left, last);
    qsort(v, _____(10)_____);
    qsort(v, _____(11)_____);
}

main()
{
    char *str[]={"111","22","3"};
    int i;

    qsort( _____(12)_____);
    for(i=0;i<3;i++)   printf("%s\n",str[i]);
}
```

Section 4: Algorithms design (10 marks for each item, total 20 marks)

1. Given a linked list (no cycle in the list), remove the ***nth*** node from the end of list and return its head. Please (1) describe your algorithm of this problem, and (2) give your function ***ListNode *removeNthFromEnd(ListNode *head, int n).***

For example,

Given linked list: 1->2->3->4->5, and n = 2.

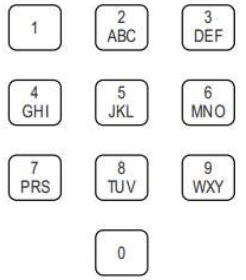
After removing the second node from the end, the linked list becomes 1->2->3->5.

You must travel the linked list only once.

Definition of the linked list:

```
typedef struct _ListNode {
    int val;
    struct _ListNode *next;
} ListNode;
```

2. On a standard telephone keypad, the digits are mapped onto the alphabet (minus the letters Q and Z) as shown in this diagram.



In order to make their phone numbers more memorable, service providers like to find numbers that spell out some word appropriate to their business that makes that phone number easier to remember. Such words that help you remember some other data are called ***mnemonics***.
In the following program, The function ***ListMnemonics*** will generate all possible letter combinations that correspond to a given number, represented as a string of digits.
Write down the definition of recursive function ***ListMnemonics*** to complete the program.

```
#include <stdio.h>
#include "genlib.h"
#include "strlib.h"
#include "simpio.h"

/* Private function prototypes */
static void ListMnemonics(string str);

/*data structure*/
static string dialpad[] = { /*map table from dial-pad to their letters */
        "0", "1", "ABC", "DEF", "GHI", "JKL", "MNO", "PRS", "TUV", "WXY"
};
static char Mnemonics[100] = {'\0'}; /*Store the mnemonics string */
static int index = 0;   /*Record the current letter position of the mnemonics */

main()
{
    string str;
    printf("This program lists all Mnemonics of a telephone number.\n");
    printf("Enter a telephone number string: ");
    str = GetLine();
    ListMnemonics(str);
}

static void ListMnemonics(string str)
{
    ……
}
```

浙江大学 2014－2015 学年夏季学期

《C 程序设计专题》课程期末考试试卷

课程号：__211Z0050___， 开课学院：__计算机学院__

考试试卷：√A 卷、B 卷（请在选定项上打√）

考试形式：√闭、开卷（请在选定项上打√），允许带__/__入场

考试日期：__2015__年__07__月__03__日，考试时间：__120__分钟

诚信考试，沉着应考，杜绝违纪.

考生姓名：_____学号：_____所属院系：_____

(注意：答题内容必须写在答题卷上，写在本试题卷上无效)

Section 1: single Choice(2 marks for each item, total 14 marks)

1. Suppose that five numbers are pushed into a stack in the order of ***1, 2, 3, 4, 5,*** and the first popped number is ***4***. What is the possible last popped number?___.
   A. 1 B. 5 C. 1 *or* 5 D. 1 *or* 3

2. In a single linked list with ***N*** nodes, which operation requires the time complexity of ***O(N)***?___. (Given: the time complexity of the operation is linear.操作时间复杂性是线性的)
   A. to find the ***i-th*** node in the list （1≤i≤N）
   B. to insert a new node after the node pointed by ***p***
   C. to arrange the nodes of the list in increasing order
   D. to delete a node after the node pointed by ***p***

3. To merge two sorted linked lists both of ***n*** nodes and return a new sorted list. What is the minimum number of compare times (最少比较次数) during the merge?___.
   A.1 B. n C. 2n D. nlogn

4. Given the following definitions, which one is the correct reference (引用)?___.
```
struct {
    int a;
    float b;
} data, *p=&data;
```
   A．(*p).data.a; B．(*p).a; C．p->data.a; D．p.data.a;

5. For the following recursive function, the return value of function call ***f(4)*** is ___.
```
int f(int n)
{
    return f(n-1)+n;
}
```
   A. 10 B. 11 C. 0 D) None of the above

6. After executing the following code fragment, the value of variable ***z*** is ___.
```
static struct {
    int x, y[3];
} a[3]={{1,2},{3,4,5},{6,7,8,9}}, *p=a+2;
int z;
z=*((int *)p-2);
```
   A. 5 B. 1 C. 7 D. None of the above

7. Given the definitions:
```
void  f1();
void (*pf)() ;
void  f2(void (*p)(), int x);
```
   In the following statements, ___ is **WRONG.**
   A. pf = f1; (*pf)(); B. pf = f1; f2(pf, 2); C. pf = f2; (*pf)(); D. f2(f, 3);

Section 2: Read the following problems and answer questions (6 marks for each item, total 30 marks)

1. Write down the definitions:
   (1) Use typedef to define a new type name ***STRPA*** _____and make it denote(表示) a pointer array of with ***10*** elements each of which points to a character(10 个元素的字符指针数组).
   (2) For the function:***int f(void *p, double a, double b);***, define a variable ***pf*** _____ to which can be assigned with the value ***f.***

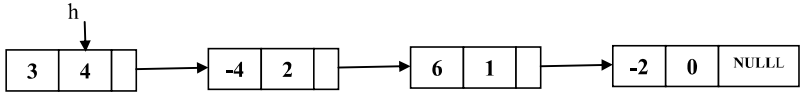
2. The following program will output _____.
```
#include <stdio.h>

void printa(int n)
{
    if(n/10)   printa(n/10);
    putchar(n%10+'0');
}

void printb(int n)
{
    putchar(n%10+'0');
    if(n/10)    printb(n/10);
}

main()
{
    printa(123);
    printb(123);
}
```

3. The following program will output _____.
```
#include <stdio.h>
#include <string.h>

main()
{
    static char t[]="REDCAP";
    void fun(char *t);

    fun(t);
```

```
    printf(t);
}

void fun(char *t)
{
    char ch;
    int i, j, n;

    n = strlen(t);
    for (i = 1; i < n; i++) {
        ch = t[i];
        for (j = i-1; j >= 0 && ch < t[j]; j--) t[j+1] = t[j];
        t[j+1] = ch;
    }

    return;
}
```

4. Given the definition of a linked list and a practical example (linked list ***h***), please draw up(画出) the result of the linked list ***h*** after calling the function ***h=f(h)***.

```
struct node{
    int coe;
    int exp;
    struct node *next;
} ;
typedef  struct node  ListNode;
```




```
ListNode *f(ListNode *head)
{
    ListNode HEAD, *cur, *pre;
    HEAD.next = head;
    cur = head;
    pre = &HEAD;
    if (!head) return NULL;
    while (cur)  {
        if(cur->coe<0) {
            pre->next= cur->next;
            free(cur);
            cur = pre->next;
        } else {
            pre = cur;
            cur = cur->next;
        }
    }
    return HEAD->next
}
```

5. Merge two sorted linked lists and return it as a new sorted list. The new list should be made by splicing (拼接) together the nodes of the original two lists. The sorted list is in ascending order (升序). For the sorted linked list ***l1*** and ***l2***

l1: 1->3->5->7->9
l2: 2->4->6

(1) Following is a wrong program. According to this program, how is the merged linked list look like after calling ***mergeTwoLists(l1,l2)***?
(2) Why is the program wrong? (Explain the reason. No need the correct program.)

```
ListNode *mergeTwoLists(ListNode *l1, ListNode *l2)
{
    ListNode HEAD;
    ListNode *cur = &HEAD;

    while (l1 && l2) {
        if (l1->val > l2->val) {
            cur->next = l1;
            l1= l1->next;
        } else {
            cur->next = l2;
            l2 = l2->next;
        }
        cur = cur->next;
    }
    return HEAD.next;
}
```

Section 3: According to the specification, complete each program (3 marks for each blank, total 30 marks)

1. Given a sorted linked list in ascending order（升序）, to insert a node with data ***x*** to this linked list and keep the order ascending.

```
ListNode *Insert( ListNode *head, int x)
   /*insert a node of x to an ascending order list head */
{
    ListNode *p, *p1=head, *p2;

    p=(ListNode *)malloc(___(1)____);
    p->data=x;
    p->next=NULL;
    while (p1 && p->data>p1->data) {
        p2=p1;
        p1=___(2)___;
    }
    if (p1==head)  head=____(3)_____;
    else p2->next=p;
    p->next=p1;
    return  ____(4)_____;
}
```

2. Given the following functions which are related with the timer(定时器) in the graphics library of our course：

```
typedef void (*TimerEventCallback) (int timerID);
void registerTimerEvent(TimerEventCallback callback);
void startTimer(int timerID, int timeinterval);
void cancelTimer(int timerID);
```

The code fragment(代码片段) below will implement the timing (定时) task with 5 seconds interval. The string "***hello***" will be displayed ONLY ONCE at the first 5 seconeds interval after starting the timer, then no more timer events happen.

Please fill in the blanks.

```
void timerEventCallback(int timerID)
{
    printf("hello\n");
    if ( timerID == 0 ) {
        _____(5)_____;
    }
}

void SetUp()
{
    ......
    registerTimerEvent(____(6)_____);
    ____(7)_____;
    ......
}
```

3. The following code fragment will implement the task that executes the function ***a***, ***b*** and ***c*** according to the number (from ***0*** to ***2***) input by the user, with ***0*** for ***a***, ***1*** for ***b***, and ***2*** for ***c***.
Please fill in the blanks to complete the code fragment.
NOTE: only ***ONE*** statement or expression can be written down in a blank.

```
void a(void);
void b(void);
void c(void);

void cmd()
{
    void (*CMDS[])(void) = {
        ____(8)_____
    };
    int k;

    scanf("%d", &k);
    if ( k >=0 && k < ____(9)_____)
    {
        ____(10)_____;
    }
}
```

Section 4: Algorithms design (13 marks for each item, total 26 marks)

1. The recursive function ***void ListPermutations(int a[],int n)*** will implement the tasks that list all the permutations(排列) with the first ***n*** objects of array ***a***, and print out these permutations.
For example, for definition ***int a[5]={11,22,33,44,55},*** the function call
ListPermutations(a, 3);
will output:

```
11  22  33
11  33  22
22  11  33
22  33  11
33  22  11
33  11  22
```

Please write down the recursive function ***ListPermutations***.

2. A rational number(有理数) is one that can be expressed as the quotient of two integers. Thus, the number 1.25 is a rational number because it is equal to 5 divided by 4. A rational number ***r=num/den*** can be represented as a structure of these two integers, thus we can define a rational type ***rationalT***. We are now going to write a program to calculate the average (平均数) of a serial rational numbers ***r1, r2, r3, …*** For example the average of ***1/2***, ***1/6***, ***3/6***, ***-5/10*** is ***1/6***. Here, ***10/60*** should be simplified to ***1/6***.

(1) Give the declaration of rational type ***rationalT***;

(2) Give the function ***rationalT simplifyRational(rationalT r)***, to simplified a rational number to its lowest term, such as simplified ***10/60*** to ***1/6***. (有理数化简函数)

(3) Give the function ***rationalT rationalsAverage(rationalT r[], int n)***, to calculate the average of ***n*** rational numbers stored in an array ***r***.

浙江大学 2015－2016 学年夏学期

《C 程序设计专题》课程期末考试试卷

课程号：__211Z0050___， 开课学院：__计算机学院__

考试试卷：√A 卷、B 卷（请在选定项上打 √）

考试形式：√闭、开卷（请在选定项上打 √），允许带__/__入场

考试日期：__2016__年__06__月__28__日，考试时间：__120__分钟

诚信考试，沉着应考，杜绝违纪.

考生姓名：_____学号：_____所属院系：_____

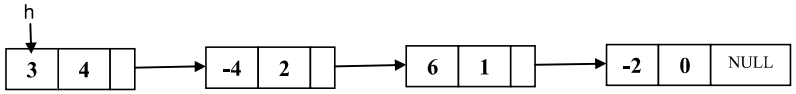
(注意：答题内容必须写在答题卷上，写在本试题卷上无效)

Section 1: Single Choice(2 marks for each item, total 20 marks)

1. Given that the pushing sequence of a stack is ***{1, 2, …, n}*** and the popping sequence is ***{ p1, p2, …, pn }***. If ***p2=n***, how many different possible popping sequences can we obtain? ___.
   A. 1 B. 2 C. n-1 D. n
2. Let ***P*** stands for push and ***O*** for pop. When using a stack to calculate the value of the postfix expression ***1 2 3 + * 4 –***, the stack operation sequence is ___.
   A. PPPOOPOO B. PPOOPPOOPPOO
   C. PPPOOPOOPPOO D. PPPOOPOOPPOOPO
3. For the following declaration, which is the correct reference to ***a***?___.
```
struct {
    int a;
    float b;
} data, *p=&data;
```
   A. (*p).data.a B. (*p).a C. p->data.a D. p.data.a
4. If a function is declared as:
   int (*func(int)) (double);
   The return type of this function is:___.
   A. An int;
   B. A pointer to an int;
   C. A pointer to a function that returns an int;
   D. A pointer to a double.
5. Given code fragment below:
```
#define SQ(x)   x*x
#define DD(x,y)  SQ(x)-SQ(y)
printf("%d", DD(2*3, 2+3);
```
   The output will be ___.
   A. 43 B. 11 C. 25 D. None of the above
6. After executing the following code fragment, the value of variable ***z*** is ___.
```
static struct {
    int x, y[3];
} a[3] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}}, *p=a;
int z;
z=*((int *)(p+1)+2);
```
   A. 3 B. 7 C. 10 D. None of the above
7. Which one of the following algorithms is ***NOT*** an ***O(n)*** algorithm?___.
   A. Finding someone in your telephone book;
   B. Linear Search;
   C. Deletion of a specific element in a double-linked List (unsorted);
   D. Comparing two strings.
8. Which one of the following algorithms is ***NOT*** an ***O(1)*** time complexity algorithm?___.
   A. Calculating the average value of the first three elements of a double-linked list;
   B. Searching in a stack;
   C. Accessing to the third element of a single-linked list;
   D. Accessing to the third element of an array.
9. Binary search uses at worst __①__, at average __②__ , and at best __③__comparisons.
   A. ①=O(log n),②=O(log n) and ③=O(1);
   B. ①=O(n), ②=O(log n) and ③=O(log log n);
   C. ①=O(n), ②=O(log n) and ③=O(1);
   D. ①=O(log n), ②=O(log n) and ③=O(log n).
10. When ***dsp("12")*** is called, the function prints out _____.(The ASCII value of '0' is 48.)
```
void dsp(char *s)
{
    if(*s)  dsp(s+1);
    printf("%d",*s);
}
```
    A. 21 B. 12 C. 5049 D. 05049

Section 2: Read the following problems and answer questions (6 marks for each item, total 30 marks)

1. Given the definition of a linked list and a practical example (linked list ***h***), Please give the value of variable ***res*** after calling the function ***res=f(h,2)***.
```
struct node {
    int coe;
    int exp;
    struct node *next;
} ;
typedef  struct node  ListNode;
```




```
int f(ListNode *h, int n)
{
    ListNode *p=h;
    int res=0, last, cur, i;

    if (h==NULL) return res;
    last=h->exp;
    while (p!=NULL) {
        cur=p->exp;
        for (i=last; i>cur; i--)  res=res*n;
        res += p->coe;
        last=cur;
        p=p->next;
    }
    for (i=last; i>0; i--)  res=res*n;
    return res;
}
```

2. For the structure declaration below, please give the values of each following expression (Note: These expressions are INDEPENDENT.):
   (1) *(++p->s) (2) ++p->x (3) (p+1)->x

```
struct {
    int x;
    char *s;
} A[2]={{1, "ab"}, {3, "cd"}}, *p=A;
```

3. Given two source code files below:

a.c:

```
#include <stdio.h>
void a() { printf("a");}
void b() { printf("b");}
void c() { printf("c");}
```

b.c:

```
#include <stdio.h>
void a();
void b();
void c();
int main()
{
    void (*CMDS[])() = {a, b, c};
    int k;
    scanf("%d", &k);
    if ( k >=0 && k < sizeof(CMDS)/sizeof(CMDS[0])) CMDS[k]();
}
```

Put them togther in one project and compile and build the executable program. When input: ***2<ENTER>***, the result to run the program is _____.

4. When input: ***3 4 8 6 7 5 9 10 2 1 <ENTER>***, the following program will print out _____.

```
#include <stdio.h>
#define NMAX 8
int getIntArray(int a[], int nmax, int sentinel)
{
    int n = 0, temp;
    do {
        scanf("%d", &temp);
        if(temp==sentinel||n==nmax)break;
        a[n++] = temp;
    }while(1);
    return n;
}

void bs(int a[], int n)
{
    int lcv,temp,lastChange,limit = n-1;
    while (limit){
        lastChange = 0;
        for (lcv=0;lcv<limit;lcv++)
            if (a[lcv]>a[lcv+1]) {
                temp = a[lcv];
                a[lcv] = a[lcv+1];
                a[lcv+1] = temp;
                lastChange = lcv;
            }
        limit = lastChange;
    }
}

int main(void)
{
    int x[NMAX],i;
    int num = getIntArray(x, NMAX, 0);
    bs(x+num/4,num/2);
    for(i=0;i<num;i++)  printf("%d#", x[i]);
    return 0;
}
```

5. When input: ***2 1<ENTER>***, the output of the following program is _____.

```
#include<stdio.h>
int ack(int m,int n)
{
    int num;
    if(m == 0) return n+1;
    if(m>0 && n==0) {
        num = ack(m-1, 1);
        return num;
    }
    num = ack(m-1, ack(m, n-1));
    return num;
}
int main(void)
{
    int m, n;
    scanf("%d %d", &m, &n);
    printf("%d", ack(m, n));
    return 0;
}
```

Section 3: According to the specification, complete each program (3 marks for each blank, total 30 marks)

1. For linked list ***h***, function ***struct node *process(struct node *h, int n, int m)*** deletes all of the nodes which data value is in the range ***[n, m]***; and function ***void printList(struct node *h)*** print all nodes' data in the linked list ***h***.Please complete the following code fragment.

```
struct node {
    int data;
    struct node *next;
};

struct node *process(struct node *h, int n, int m)
{
    struct node *p, *q;
    p=__(1)__;
    while (p!=NULL)  {
        if (p->data >=n && p->data <=m) {
            if (p==h) { p=p->next;  free(h);  h=p; }
```

```
            else {
                q->next=___(2)___;
                free(p);
                p=q->next;
            }
        } else  {
            q=__(3)__;
            p=p->next;
        }
    }
    return ____(4)____;
}

void printList(struct node *h)
{
    struct node *p=h;

    while (p!=NULL) {
        printf("%d " , p->data);
        ____(5)_____;
    }
}
```

2. The timer and char functions in the course graphics library are:

typedef void (*CharEventCallback) (char c);
typedef void (*TimerEventCallback) (int timerID);
void registerCharEvent(CharEventCallback callback);
void registerTimerEvent(TimerEventCallback callback);
void startTimer(int id,int timeinterval);
void cancelTimer(int id);

The code fragment below is to display ***"hello"*** every ***five seconds*** for ***three times*** when the space bar is pressed. Please fill in the blanks below.

```
void key_pressed(char c);
void timer_touch(int id);

void SetUp()
{
        …...
        registerCharEvent(____(6)_____);
        registerTimerEvent(____(7)____ );
        …...
}

void key_pressed(char ch)
{
        if ( ch == ' ' ) {
                ____(8)____;
        }
}

void timer_touch(int id)
{
        ___(9)___ int count = 0;
        if ( id == 0 ) {
                printf("hello\n");
                if ( ++count ==3 ) {
                        _______(10)_______;
                }
        }
}
```


Section 4: Algorithms design (10 marks for each item, total 20 marks)

1. A string consists of brackets (括号，含{,},[,],(,)). We can use stack to check whether these brackets are matching. For examples, "{[]()}" is a matching string, but "{[()}]" is not. Please:

(1) According to the following declarations, complete the stack's operation functions ***Push()*** and ***Pop()***.

```
#define MAXSIZE 100
struct Stack {
   char S[MAXSIZE];
   int top;
};
typedef struct Stack *StackP;

StackP CreatStack()
{
   StackP *sp;
   sp=(StackP)malloc(sizeof(struct Stack));
   sp->top= -1;
   return sp;
}

void Push(StackP sp, char c)
{…….}

char Pop(StackP sp)
{…. }
```

(2) Complete the function ***int Check(char *BracketsStr)***, to check whether the brackets in the string ***BracketsStr*** are matching. If the brackets are matching, return ***1***, else return ***0***.

2. Given a polynomial $f_n(x)=a_0+a_1x+a_2x^2+\ldots+a_nx^n$, for a given x, if calculate the value separately for each item, $1+2+\ldots+n=n(n+1)/2$ times multiplications will be needed and the efficiency is very low. If rewrite the formula as:

$$f_n(x)=((\ldots(((a_n)x + a_{n-1})x + a_{n-2})x +\ldots)x + a_1)x + a_0$$

a recursive algorithm can be used to calculate the value of polynomial function $f_n(x)$ more efficient.

(1) Please design the recursive algorithm of calculating the n-order polynomial $f_n(x)$, including the data structure and the two recursion-conditions.

(2) Write down the function of implementing the recursive algorithm above, and analyze the required number of multiplications.

浙江大学 2016－2017 学年夏学期

《C 程序设计专题》课程期末考试试卷

课程号：__211Z0050___， 开课学院：__计算机学院__

考试试卷：√A 卷、B 卷（请在选定项上打√）

考试形式：√闭、开卷（请在选定项上打√），允许带__/__入场

考试日期：__2017__年__06__月__29__日，考试时间：__120__分钟

诚信考试，沉着应考，杜绝违纪.

考生姓名：_____学号：_____所属院系：_____

(注意：答题内容必须写在答题卷上，写在本试题卷上无效)

Section 1: single Choice(2 marks for each item, total 20 marks)

1. Suppose that a stack is stored in an array ***V[N]***. If the stack is empty, the initial ***top*** pointer(初始栈顶指针) is ***N***. When push an element ***x*** into the stack, the correct operation is___.
   A．top=top+1;V[top]=x;　　B．V[top]=x;top=top+1;
   C．top=top-1;V[top]=x;　　D．V[top]=x;top=top-1;
2. Using a ring buffer to represent a queue(循环队列). The queue is stored in an array ***Q[MAXSIZE]*** with the head pointer ***front*** and the tail pointer ***rear***, where ***front*** points to the first element and ***rear*** to the position next to the last element of the queue. At the beginning the queue is empty, ***front*** is ***-1*** and ***rear*** is ***0***. When add a new element ***x*** to the queue, the correct operation is_____.(Suppose that the queue is ***NOT*** full.)
   A. Q[front] = x; front = front - 1;
   B. Q[front] = x; front = (front - 1) % MAXSIZE;
   C. Q[rear] = x; rear = rear + 1;
   D. Q[rear] = x; rear = (rear + 1) % MAXSIZE;
3. Given a stack ***S*** and a queue ***Q*** both with the EMPTY initial status. Elements ***e1***、***e2***、***e3***、***e4***、***e5***、***e6*** loop through(依次通过) the statck ***S***. Once an element is popped out of the stack, it is added to the queue ***Q*** immediately. If the 6 elements leave the queue ***Q*** with the order of ***e2***、***e4***、***e3***、***e6***、***e5***、***e1***, the size of the stack ***S*** is ___ AT LEAST(至少).
   A. 2　　B. 3　　C. 4　　D. 6
4. The statement ___ below is WRONG.
   A. The variable names in C must be lowercase(小写) and constants uppercase.
   B. The C program runs by editing, compiling, linking, and executing.
   C. The three basic structures of the C language are sequence, branch and loop.
   D. The C programs must be made up of functions.
5. The compilers(编译器) usually use the ___ data structure to process the recursive function-calling(递归函数调用).
   A. queue　　B. array　　C. stack　　D. record

6. Given a simple linked list and a pointer ***p*** points to a node. Executing the statements _____ below will delete the successor(后继结点) of the node which ***p*** points to.
   A. p=p->next; free(p);
   B. p->next=p->next->next; free(p);
   C. q= p->next ;q->next=p->next; free(q);
   D. q=p->next; p->next=q->next ;free(q);
7. Which of the followings should ***NOT*** be placed in the header file?___ .
   A．function prototype(原型)　　B．function definition
   C．Macro(宏) definition　　D．data-type definition
8. Given the definitions：
   #define type1 char *
   typedef char *type2;
   type1 s1, s2;
   type2 s3, s4;
   Among the followings, which group contains all the identifiers that are character pointers? ___.
   A．s1, s3　　B．s1, s2, s3
   C．s1, s3, s4　　D．s1, s2, s3, s4
9. Inserting a node into an ascending-order(升序) linked list with ***n*** nodes needs _____ comparisons at average.
   A. O(n)　　B. O(n log n)　　C. O(n^2)　　D. O(1)
10. After executing the following code fragment, the value of variable ***z*** is _____.
```
static struct{
    int x, y[3];
} a[3]={{0},{5,6,7},{10,12}}, *p=a+3;
int z;
z=*((int *)(p-1)-3);
```
   A. 0　　B. 10　　C. 6　　D. None of the above

Section 2: Read the following problems and answer questions (5 marks for each item, total 30 marks)

1. Fill in the blanks.
   (1) When calling ***printf("%d",fun("2017-06-29"))***, the output is _____.
```
int fun(char *str)
{
   if(*str == 0) return 0;
   return fun(str+1)+1;
}
```
   (2) Use ***typedef*** to define a new type name ***KeyboardEventCallback*** _____which can be as the pointer type of keyboard event call-back function (键盘事件回调函数) .NOTE: the prototype of keyboard event call-back function is ***void KbEventProc(int, int)***.
2. The following program will output_____.
```
#include <stdio.h>

int f1(int (*f)(int));
int f2(int i);

int main(void)
{
    printf("Answer: %d", f1(f2));
    return 0;
}
```

```
int f1(int (*f)(int))
{
    int n=0;
    while ( (*f)(n) ) n++;
    return n;
}

int f2(int i)
{
    return i*i+i-12;
}
```

3. Odd-Even Sort / Brick Sort, as follows, is basically a variation of bubble sort(冒泡排序). This algorithm is divided into two phases-Odd and Even Phase. The algorithm runs until the array elements are sorted and in each iteration two phases occurs-Odd and Even Phases.

```
void oddeven_sort(int arr[], int n)
{
   char is_sorted = 0;
   int i;
   while (!is_sorted)  {
      is_sorted = 1;
      for (i=1; i<=n-2; i=i+2) {
         if (arr[i] > arr[i+1]) {
            swap(arr,i, i+1);/*exchange the elements of index i and i+1*/
            is_sorted = 0;
         }
      }
      for (i=0; i<=n-2; i=i+2)  {
         if (arr[i] > arr[i+1])  {
            swap(arr, i, i+1);
            is_sorted = 0;
         }
      }
   }
   return;
}
```

Please describe: (1) one of the best case for this algorithm and (2) provide the best-case performance (computational complexity).

4. Given the definition of a linked list and a practical example (list ***h***), please write out the result of the linked list after calling the function ***f(h,4)***.

```
struct ListNode {
      int val;
      struct ListNode *next;
};
struct ListNode *f(struct ListNode *head, int x)
{
      struct ListNode *root = (struct ListNode *)malloc(sizeof(struct ListNode));
      struct ListNode *pivot = (struct ListNode *)malloc(sizeof(struct ListNode));
      struct ListNode *root_last = root;
      struct ListNode *pivot_last = pivot;
      struct ListNode *current = head;

      memset(root, 0, sizeof(struct ListNode));   /*Initial root spaces as zeroes*/
      memset(pivot, 0, sizeof(struct ListNode)); /* Initial pivot spaces as zeroes */
      while (current != NULL) {
         if (current->val < x) {
            root_last->next = current;
            root_last = current;
         } else {
            pivot_last->next = current;
            pivot_last = current;
         }
         current = current->next;
      }
      root_last->next = pivot->next;
      pivot_last->next = NULL;
      return root->next;
   }
```

The input linked list ***h*** is:
1->4->3->2->5->2 and x = 4.

5. The following program will output_____.

```
#include <stdio.h>
void fun(int num, int n)
{
   if(num > n) {
      while (num % n)  n++;
      num /= n;
      printf("%d#", n);
      fun(num,n);
   }
}
int main()
{
   int n = 1001;
   fun(n, 2);
   return 0;
}
```

6. Please describe the function(功能) of the following program _____.
Note: the following program is based on our course's graphics library.

```
#include <windows.h>
#include <winuser.h>
#include "graphics.h"
#include "extgraph.h"
#include "genlib.h"
double ccx, ccy;
double radius = 1.0;
double mx, my,omx , omy ;
void DrawCenteredCircle(double x, double y, double r);
bool inBox(double x0, double y0, double x1, double x2, double y1, double y2);
void MouseEventProcess(int x, int y, int button, int event);

void Main()
{
    InitGraphics();
    registerMouseEvent(MouseEventProcess);
    ccx = GetWindowWidth()/2;ccy = GetWindowHeight()/2;
    DrawCenteredCircle(ccx, ccy, radius);
}
```

```
void DrawCenteredCircle(double x, double y, double r)
{
    MovePen(x + r, y);
    DrawArc(r, 0.0, 360.0);
}

bool inBox(double x0, double y0, double x1, double x2, double y1, double y2)
{
    return (x0 >= x1 && x0 <= x2 && y0 >= y1 && y0 <= y2);
}

void MouseEventProcess(int x, int y, int button, int event)
{
    static bool isDraw = FALSE;
    mx = ScaleXInches(x);/*pixels --> inches*/
    my = ScaleYInches(y);/*pixels --> inches*/
    switch (event) {
      case BUTTON_DOWN:
        if (button == LEFT_BUTTON) {
          if (inBox(mx,my,ccx-radius,ccx+radius,ccy-radius, ccy+radius))
            isDraw = TRUE;
        }
        omx = mx; omy = my;
        break;
      case BUTTON_UP:
        if (button == LEFT_BUTTON) isDraw = FALSE;
        break;
      case MOUSEMOVE:
        if (isDraw) {
          SetEraseMode(TRUE);
          DrawCenteredCircle(ccx, ccy, radius);
          ccx += mx - omx; ccy += my - omy;
          omx = mx; omy = my;
          SetEraseMode(FALSE);
          DrawCenteredCircle(ccx, ccy, radius);
        }
        break;
    }
}
```

Section 3: According to the specification, complete each program (2 marks for each blank, total 30 marks)

1. The following counting sort(计数排序) algorithm assumes that each of the items is an integer in the range ***0*** to ***key_range-1***, for some integer ***key_range***. It loops over the items, computing a histogram(直方图) of the number of times each key occurs within the input collection array. It then performs a prefix sum computation to determine, for each key, the starting position in the output array of the items having that key. Finally, it loops over the items again, moving each item into its sorted position in the output array. The final output is as follows.

    4 9 1 8 7 6 6 5 9 3 2 1 0
    0 1 1 2 3 4 5 6 6 7 8 9 9

    Please complete it.

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



```
const int key_range=10;
int array[]={4, 9, 1, 8, 7, 6, 6, 5, 9, 3, 2, 1, 0};
const int num = sizeof(array)/sizeof(int);

void print_output(int output_array[], int num)
{
   int i;
   for(i=0; i<num; i++) printf("%d ", output_array[i]);
   printf("\n");
   return;
}

void count_sort(int input_array[], int num)
{
   int total, i, count[key_range];
   int *output_array = (int*) malloc(_____(1)_____);
   if(!output_array){
      fprintf(stderr, "Out of memory.\n");
      return;
   }
   /* Initialize. */
   memset(count, 0, sizeof(int)*key_range);

   /* Collect statistics. */
   for(i=0; i<num; i++) count[input_array[i]]++;

   /* Calculate the start position. */
   for(_____(2)_____,i=0; i<key_range; i++){
      int oldcount = count[i];
      count[i] = total;
      total += oldcount;
   }

   /* Rerange. */
   for(i=0;_____(3)_____; i++){
      output_array[count[input_array[i]]]= input_array[i];
      _____(4)_____;
   }

   /* Write back. */
   for(i=0; i<num; i++)  input_array[i]=output_array[i];

   return;
}

int main(void)
{
   print_output(array, num);
   count_sort(_____(5)_____);
   print_output(array, num);
   return 0;
}
```

2. A stack is usually stored in an array with a stack top pointer(栈顶指针) which is the index of the array. When the stack is empty, the stack top pointer is **-1.** The following code fragment implement(实现) the basic stack operations: ***push*** and ***pop.*** Please fill in the blanks to complete the program.

```
#define MAX_STACK 1000
int  Stack[MAX_STACK];
int sp = _____(6)____; /*stack top pointer*/

void  push(int  n)
{
     if (sp == _____(7)____)  return; /*stack is full*/
     Stack[ ___(8)____] = n; /*store n to the array*/
}

int  pop(void)
{
     if (_____(9)____)  return  NULL; /*stack is empty*/
     return _____(10)____;
}
```

3. Given: a simple linked list ***A*** and ***B*** represent tow sets(集合). ***A*** and ***B*** are the head pointers respectively. The following code fragment implement ***A=A-B.*** For example: If linked list ***A*** is: ***1->4->3->2->5*** and ***B*** is : ***1->3***, after executing the calling ***A=setminus(A,B)***, the list ***A*** is: ***4->2->5.*** Please complete the program.

```
#include <stdio.h>
 typedef struct node {
    int data;
    struct node *next;
 } lnode;

 lnnode *setminus(lnode *A, lnode *B)
 {
    lnode *p,*q;

    if (A == NULL || B == NULL) return __(11)__;
    while (B != NULL) {
       p = A;
       if (p == NULL) break;
       while (p ->data != B->data && _____(12)_____) {
          q = p; p = p->next;
       }
       if (_____(13)_____) {
          if (p == A) {
               A = p->next;
          } else {
               _____(14)_____;
               free(p);
          }
       }
       _____(15)_____;
    }
    return A;
 }
```


Section 4: Algorithms design (20 marks for each item, total 20 marks)

1. Given an array and a number ***k*** where ***k*** is smaller than size of array. we need to design an algorithm with computational complexity of ***O(nk)*** to find the ***k'th*** smallest element in the given array. For examples:
   Input: arr[] = {7, 10, 4, 3, 20, 15}, k = 3, Output: 7;
   Input: arr[] = {7, 10, 4, 3, 20, 15}, k = 4, Output: 10.
   Please write a function to implement the algorithm.

```
void ksort(int arr[], int n, int k) /*n is the number of elements of array arr */
{
   ……
}
```

2. Given a set(集合), such as ***{1, 2, 3}***, print out all the subsets(子集合). A subset refers to the collection of any of the elements in the original set. For example, all the subsets of set ***{1,2,3}*** are: ***{},{1},{2},{3},{1,2},{1,3},{2,3}*** and ***{1,2,3}.***
   Please write a ***RECURSIVE*** function to resolve the problem. NOTE: the set is stored in an array, and it's subsets can be displayed in ***ANY*** order(任意顺序).
   Tips: For any element in the original set, there are two options: ***INCLUDING*** or ***NOT***. You can define a boolean array ***flag[]*** to store the corresponding states of every elements of the set. First, determine the first element's state,TURE or FALSE, then process the left elements with the same method.

```
#include <stdio.h>
#define N 100
#define TRUE  1
#define FALSE 0
typedef int bool
void RecursiveSubsets(int set[], bool flag[], int start, int end);

int main()
{
   int set[] = {1, 2, 3};
   int n = sizeof(set) / sizeof(set[0]);
   bool *flag= (bool *)malloc(sizeof(bool)*n);

   RecursiveSubsets(set, flag, 0, n-1);
}

void RecursiveSubsets(int set[], bool flag[], int start, int end)
{/*start and end are both indexes of array*/
   ……
}
```

浙江大学 2017－2018 学年夏学期

《C 程序设计专题》课程期末考试试卷

课程号：__211Z0050__， 开课学院：__计算机学院__

考试试卷：√A 卷、B 卷（请在选定项上打√）

考试形式：√闭、开卷（请在选定项上打√），允许带__/__入场

考试日期：__2018__年__07__月__05__日，考试时间：__120__分钟

诚信考试，沉着应考，杜绝违纪.

考生姓名：_____学号：_____所属院系：_____

(注意：答题内容必须写在答题卷上，写在本试题卷上无效)

Section 1: single Choice(2 marks for each item, total 20 marks)

1. Given the definitions as below:
```
typedef struct {double x, y;} PointT;
PointT p1, p2;
int i;
char *pc1 = (char *)&p1, *pc2 = (char *)&p2;
```
In the followings, the statement _____ ***CANNOT*** copy the value of ***p1*** to ***p2*** CORRECTLY.
A. p2 = p1;
B. p2.x = p1.x; p2.y = p1.y;
C. strcpy(pc2, pc1);
D. for (i = 0; i < 16; i++) *pc2++ = *pc1++;

2. After executing the following code fragment, the value of variable ***z*** is _____.
```
static struct{
    int x, y[3];
} a[3] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}}, *p = a+1;
int z;
z = ((int *)p+1)[-2];
```
A. 4 B. 5 C. 6 D. 7

3. Six elements ***a,b,c,d,e,f*** are pushed into a stack consequently, and pop operations are performed alternately. If there are ***NO*** continuous three pop operations(连续 3 次 pop 操作), which of the followings is the impossible output? _____
A. afedcb B. cbdaef C. dcebfa D. bcaefd

4. In a linked list, which of the following is the possible statement to delete the node after ***p***? _____
A. p=p->next; B. p=p->next->next;
C. p->next=p; D. p->next=p->next->next;

5. For the following declarations, which ***scanf*** statement is wrong? _____
```
struct Student{
    char name[10];
    int age;
    char gender;
} std[3], *p=std;
```
A. scanf("%d",&(*p).age); B. scanf("%c",&std[0].gender);
C. scanf("%c",&(p->gender)); D. scanf("%s",&std.name);

6. What is the output of the following statements? _____
```
struct {
    int x, y;
} s[2] = { { 1, 3 }, { 2, 7 } };
printf("%d", s[0].y/s[1].x );
```
A. 0 B. 1 C. 2 D. 3

7. when sorting ***n*** objects, if input array is already sorted, the Bubble Sort algorithm has _____ time complexity.
A. O(1) B. O(nlogn) C. O(n) D. $O(n^2)$

8. Inserting a node into a descending-order(降序) linked list with ***n*** nodes needs _____ comparisons at average.
A. O(1) B. O(nlogn) C. $O(n^2)$ D. O(n)

9. For keyword ***static***, which one below is NOT correct? _____
A. A static local variable is persistence after calling the function.
B. A static global variable is not accessible by other compile units.
C. A static function is not accessible by other compile units.
D. "static int *p;"means that p is a pointer to a static int.

10. According to the following function-declaration, the return value is _____ after calling ***f(4)***.
```
int f(int n)
{ if (n) return f(n - 1) + n; else return n; }
```
A. 0 B. 4 C. 10 D. None of the above

Section 2: Read the following problems and answer questions (5 marks for each item, total 30 marks)

1. Write down your answers.
(1) When calling ***fun("2018-07-05");*** the output is _____.
```
void fun(char *str)
{
   if (*str) fun(str+1);
   if (*str) putchar(*str);
   return;
}
```
(2) Given a function ***fun*** which has a formal parameter of type ***int*** and returns a pointer to the function ***void f(int n)***. How to describe the prototype of ***fun***?
_____

2. When input ***23<ENTER>***, the following program will output_____.
```
#include <stdio.h>
#include <string.h>
char *dialpad[] = {
    "0","1","AB","CDE","FGH","JKL","MNO","PRS","TUV","WXY"
};
char Mnemonics[20];
int indx = 0;

int main()
{
    void ListMnemonics(char *str);
    char num[10];
    scanf("%s", num);
    ListMnemonics(num);
}
```

```
void ListMnemonics(char *str)
{
     char *ptr;
     if (strlen(str)==0) {
        Mnemonics[indx] = '\0';
        printf("%s#", Mnemonics);
     } else {
        ptr = dialpad[*str-'0'];
        while (*ptr) {
              Mnemonics[indx++] = *ptr++;
              ListMnemonics(str+1);
        }
   }
   return;
}
```

3. Execute the fllowing program, what will happen when press the key ***ESC***? _____

Note: the following program is based on our course's graphics library.

```
#include <windows.h>
#include <winuser.h>
#include "graphics.h"
#include "extgraph.h"
#include "genlib.h"
bool flag = FALSE;
double x, y;
void KeyboardEventProcess(int key,int event);
void TimerEventProcess(int timerID);

void Main()
{
   InitGraphics();
   registerKeyboardEvent(KeyboardEventProcess);
   registerTimerEvent(TimerEventProcess);
   x = GetWindowWidth()/2;
   y = GetWindowHeight()/2;
   return;
}

void KeyboardEventProcess(int key,int event)
{
   if (event==KEY_DOWN && key==VK_ESCAPE) startTimer(1, 500);
   return;
}

void TimerEventProcess(int timerID)
{
   static char buf[2] = "9";
   static int count = 0;
   static flag = TRUE;
   if (timerID == 1) {
      SetEraseMode(!flag);
      MovePen(x, y);
      DrawTextString(buf);
      if (flag = !flag) buf[0]--;
      if (++count == 20) cancelTimer(1);
   }
   return;
}
```


4. For the following program and a linked list ***h***: ***1->2->3->4->5->6->7->8***,
   (1) what will the function-call ***f(h,3)*** prints out? _____
   (2) what will the function-call ***f(h,3)*** return? _____

```
struct ListNode {
   int data;
   struct ListNode *link;
} ;
typedef struct ListNode *LinkList
LinkList h;

int f( LinkList list, int k )
{
   LinkList p, q;
   int count = 0;
   p = q = list->link;
   while ( p && count<k ) {
      p = p->link;
      count++;
   }
   if ( p ) {
      while ( p ) {
         p = p->link;
         q = q->link;
      }
   }
   if ( count < k )
      return 0;
   else {
      printf("%d\n",q->data);
      return 1;
   }
}
```

5. For the following program and a linked list ***h***: ***(-3)->(4)->(5)->(-2)->(-1)->(2)->(6)->(-2)***, after executing process(***h***), what will the program output?

```
struct node {
      int data;
      struct node *next;
};
void process(struct node *h)
{
      struct node *p, *thisp,*maxp;
      int this=0, max=0;
      p=thisp=h;  maxp=NULL;
      while (p) {
         this=this+p->data;
         if (this>max) {
            max=this;
            maxp=thisp;
         } else if (this<0) {
            this=0;
            thisp=p->next;
         }
         p=p->next;
      }
      p=maxp;
```

```
    while (p && (max>0)) {
        printf("%d " , p->data);
        max=max - p->data;
        p=p->next;
    }
    return;
}
```

6. Gnome Sort also called Stupid sort is based on the concept of a Garden Gnome sorting his flower pots(花盆). A garden gnome sorts the flower pots by the following method:
(1) He looks at the flower pot next to him and the previous one; if they are in the right order he steps one pot forward, otherwise he swaps them and steps one pot backwards.
(2) If there is no previous pot (he is at the starting of the pot line), he steps forwards; if there is no pot next to him (he is at the end of the pot line), he is done.

We can implement Gnome Sort as follows.

```
void gnome(int arr[], int n)
{
    int i = 0, t;
    while (i < n) {
        if (i == 0 || arr[i] >= arr[i-1]) {
            i++;
        } else {
            t = arr[i-1];
            arr[i-1] = arr[i];
            arr[i] = t;
            i--;
        }
    }
    return;
}
```

Please describe one of the best case for this algorithm and provide the best-case performance (computational complexity).

Section 3: According to the specification, complete each program (2 marks for each blank, total 30 marks)

1. The following program realizes the function of strokes, that is, when the left mouse button is pressed and dragged, a trace is drawn in the window along with the mouse position, and when the left mouse button is raised, the trace is not drawn. Please fill in the blanks to complete the program.
Note: the following program is based on our course's graphics library.

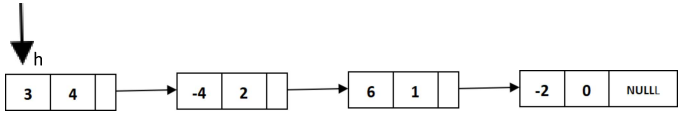
```
#include <windows.h>
#include <winuser.h>
#include "graphics.h"
#include "extgraph.h"
#include "genlib.h"

void Painter(int x, int y, int button, ____(1)____ )
{
    double cx = ScaleXInches(x);
    double cy = ScaleYInches(y);
    double dx,dy;
```



```
    static int isLeftButtonDown = 0;
    ____(2)____ lx = 0.0, ly = 0.0;

    switch(event){
        case BUTTON_DOWN:
            if (button==LEFT_BUTTON)   isLeftButtonDown = 1;
            lx = cx;    ly = cy;
            break;
        case BUTTON_UP:
            if (button==LEFT_BUTTON)  isLeftButtonDown = 0;
            break;
        case ____(3)____:
            if (isLeftButtonDown) {
                MovePen(lx,ly);
                dx = cx-lx; dy = cy-ly;
                DrawLine(dx,dy);
                lx = cx; ly = cy;
            }
            break;
    }
    return;
}

void Main()
{
    ____(4)____;
    SetPenSize(1);
    SetPenColor("Black");
    ____(5)____;
    return;
}
```

2. The following linked list is a polynomial(多项式), where each node is a term of the polynomial consisting of coe (coefficient) and exp (exponential). The following function f(h,x) is to compute the value of the polynomial h with a certain x.

```
struct node {
    int coe;
    int exp;
    struct node *next;
} ;
typedef  struct node  ListNode;
```




```
int f(ListNode *h, int x)
{
    ____(6)____ *p=h;
    int res=0, last, cur, i;

    if (h==NULL) return res;
    last=h->exp;

    while(p!=NULL) {
```

```
        cur=p->exp;
        for (i=last; i>cur; i--)
            res=      (7)      ;
        res += _____(8)_____;
        last=cur;
        p=_____(9)_____;
    }
    for (i=last; i>0; i--) res=res*x;
    _____(10)_____;
}
```

3. The Binary Insertion Sort algorithm use binary search to find the proper location to insert the selected item at each iteration, thus, it reduces the number of comparisons in normal insertion sort. The final output of our implementation should be as follows.

0 12 17 23 31 37 46 54 72 88 100

Please complete the following Binary Insertion Sort implementation.

```
#include <stdio.h>
int binarySearch(int a[], int item, int low, int high)
{
   int mid;

   if (high <= low) return (item > a[low])?  (low + 1): low;
   mid = _____(11)_____;
   if(item == a[mid])  return mid;
   if(item > a[mid])  return binarySearch(a, item, mid+1, high);
   return binarySearch(a, item, low, ____(12)____);
}

void insertionSort(int a[], int n)
{
   int i, loc, j, k, selected;

   for (i = 1; i < n; ++i)  {
      j = i - 1;
      selected = a[i];
      loc = binarySearch(a, selected, 0, ____(13)____);
      while (j >= loc) {
         a[j+1] = a[j];
         j--;
      }
      a[j+1] = _____(14)_____;
   }
   return;
}

int main()
{
   int a[] = {37, 23, 0, 17, 12, 72, 31, 46, 100, 88, 54};
   int n = __(15)__/sizeof(a[0]) , i;

   insertionSort(a, n);
   for (i = 0; i < n; i++) printf("%d ",a[i]);
   return 0;
}
```


Section 4: Algorithms design (10 marks for each item, total 20 marks)

1. For a linked list ***h***, function ***int fun(struct node *h)*** is to determine whether the linked list ***h*** is central symmetry（中心对称）. If it is, return 1, or return 0. E.g. ***1->2->3->2->1*** and ***1->2->2->1*** is central symmetry, and ***1->2->1->2*** is not. Please implement the function ***fun(struct node *h)*** with a ***STACK***. You need not to implement the function ***push()*** and ***pop()***. You can call the functions directly: ***void push(int n), int pop()***.

```
struct node {
      int data;
      struct node *next;
};

int process(struct node *h)
{
      ……
}
```

2. Given an array of numbers, push all the zeros to the end of the array. For example, if the given arrays is ***{1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0, 9}***, it should be changed to ***{1, 9, 8, 4, 2, 7, 6, 9, 0, 0, 0, 0}***. The order of all other elements should be same. Expected time complexity is ***O(n)*** and extra space is ***O(1)***.

Assuming we already implement the main function as follows.

```
int main()
{
   int arr[] = {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0, 9};
   int n = sizeof(arr) / sizeof(arr[0]);
   void push_zeros_end(int arr[], int n);

   push_zeros_end(arr, n);
   for (int i = 0; i < n; i++) printf("%d ", arr[i]);
   return 0;
}
```

Please implement the ***push_zeros_end*** function.

```
void push_zeros_end(int arr[], int n)
{
      ……
}
```