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*/
    ……
}
```