浙江大学 2020－2021 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：__2021__ 年 __07__ 月 __01__ 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. For definition "***typedef struct{char * name;} *T; T t;***", which one below is different from several other items ? _____.
   A. t->name   B. &t->name[0]   C. (*t).name   D. *t.name

2. According to the following code, which one below is CORRECT? _____.
```
#include<stdio.h>
struct decl {
    int n = 100;
} d1;
int main()
{
    printf("%d",d1.n);
    return 0;
}
```
   A. Output 0.
   B. Output 100.
   C. Output some garbage value;
   D. This program will have a compilation error.

3. In which of the following is ***p*** a pointer variable? _____.
   A. int* *p();   B. int *p();   C. int (*p)[5];   D. int *p[6];

4. In a C source file, if you want to define a global variable that only allows all functions in the source file to be used, the type modifier for that variable is _____.
   A. extern   B. local   C. static   D. auto

5. If you have a function ***max (a,b)***, and you have the function pointer variable ***p*** pointing to ***max***, the correct method to call the function is _____.
   A. (*p)max(a+b);   B. *pmax(a,b);   C. (*p)(a,b);   D. *p(a,b);

6. Which of the following statements is NOT true about recursive functions? _____.
   A.They divide larger problems into smaller ones of the same form;

B.They typically begin with a test for simple case(s);
C.They are often more efficient than non-recursive implementations of the same algorithm;
D.They can often solve complex problems in a concise(简明的) way.

7. When sorting an array ***1 2 3 4 7 5*** in ascending order(升序), which of the following algorithms is the most efficient? _____.
A. Bubble Sort B. Selection Sort C. Merge Sort D. Insertion Sort

8. Given the popping sequence of a stack as ***{a, b, c, d, e}***. Among the followings, the impossible pushing sequence is _____.
A. c b a e d B. d e a c b C. e a b c d D. e d a b c

9. Let ***h*** be the head of a singly linked list without a dummy head node. To insert a new node ***t*** as the first node, we must do: _____.
A. h=t; t->next=h->next; B. t->next=h->next; h=t;
C. h=t; t->next=h; D. t->next=h; h=t;

10. Suppose that an array of size ***m*** is used to store a circular queue. If the front position is ***front*** and the current size is ***size***, then the rear element must be at: _____.
A. front+size B. front+size-1
C. (front+size)%m D. (front+size-1)%m

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

1. Write down your answers.
(1) Assume that the operators **+**, **-**, * are left associative and **^** is right associative. The order of precedence (from highest to lowest) is ***^, *, +, -.*** The postfix expression corresponding to the infix expression ***a + b * c - d ^ e ^ f*** will be ?
____(2 marks)

(2) Using ***typedef*** to create an alias for the data type of a pointer to an array of ***20*** elements, each element is a structure which has two integer members, ***a*** and ***b***.
____(3 marks)

2. The fllowing program will output __________________.

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

int main()
{
    struct xyz {
        char *n;
        char m[20];
        int eos;
    };
    struct xyz x = {"ABC", "XYZ", 2021}, y=x;

    x.n = "DEF"; x.m[0]= ‘P’;
    printf("%s#%s#",y.n, y.m);
    return 0;
}
```

3. For linked list "***1->2->3->4->5->/***", if ***head*** is a pointor to the first node, ***foo(head, 2, 2)*** will output? ____________

```
void foo (struct Node  *head, int M, int N)
{
    struct Node *curr = head, *t;
    int count;

    for (count=1; count<M && curr!=NULL; count++)  curr = curr->next;
    if (curr == NULL)  return;
    t = curr->next;
    for (count=0; count<N && t!=NULL; count++) {
        struct Node *temp = t;
        t = t->next;
        free(temp);
    }
    curr->next = t;
    t = head;
    while (t != NULL) {
        printf("%d#", t->data);
        t = t->next;
    }
}
```

4. Given the following function definition, write down the output of calling ***hex(445)*** ____________.

```
void hex(int decimal)
{
   if (decimal) {
       hex(decimal/6);
       printf("%d", decimal%6);
   }
}
```

5. 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 following is an implementation of the algorithm.

```
#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 = (low+high)/2;
    if(item == a[mid]) return mid;
    if(item > a[mid]) return binarySearch(a, item, mid+1, high);
    return binarySearch(a, item, low, mid-1);
}

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, j);
        while (j >= loc) {
            a[j+1] = a[j];
            j--;
        }
       a[j+1] = selected;
    }
    return;
}
```

Please describe: (1) one of the best cases for this algorithm and (2) provide the best case and (3) average performance (computational complexity).

6. Given an abstact queue data type(抽象队列数据类型) ***queueADT*** and its functions below:

   *typedef struct queueCDT *queueADT; // definition of abstract queue data type*
   queueADT NewQueue(void); // Create an empty queue
   void FreeQueue(queueADT queue);//Free the space of the queue
   *void Enqueue(queueADT queue, void *obj); // Enter an element obj to queue*
   *void *Dequeue(queueADT queue);//Delete an element from queue*
   int QueueLength(queueADT queue);// Return the length of queue

   When input ***Zhejiang<ENTER>*** from the keyboard, the following program will output ________.

```
/* All necessary header files are omitted*/

int main()
{
    queueADT eQueue, dQueue;
    char *chPtr, ch;

    eQueue = NewQueue();
    dQueue = NewQueue();
    while ((ch = getchar()) != '\n') {
        chPtr = (char *)malloc(sizeof(char));
        *chPtr = ch;
        Enqueue(eQueue, chPtr);
    }
    while (QueueLength(eQueue) != 0) {
        Enqueue(dQueue, Dequeue(eQueue));
        if (QueueLength(eQueue) != 0) Enqueue(eQueue, Dequeue(eQueue));
    }
    while (QueueLength(dQueue) != 0)  putchar(*(char *)Dequeue(dQueue));
    return 0;
}
```

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

1. Assuming that we have a linked list node defined as follows:

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

Please complete the following function ***list_reverse*** which reverse a linked list. For example, for the following linked list,

    1->3->5->7->/

after execution of ***list_reverse***, the linked list should be changed to,

    7->5->3->1->/

Note that the ***head_ref*** is the address of the pointer to its first node.

```
void list_reverse(struct Node** head_ref)
{
    struct Node* prev = NULL;
    struct Node* current =  __(1)__;
    struct Node* next = NULL;

    while (current !=  __(2)__ ) {
        next = current->next;
        current->next = prev;
       __(3)__;
       current = next;
    }
    *head_ref = __(4)__ ;
}
```

2. A typical optimization of Selection Sort is to select both the minimum and the maximum elements when scanning the unsorted part of the input array, and update the sorted arrays at two ends (be careful for the special case if the maximum element is at the front of the unsorted part or the minimum element is at the end). Complete the following implementation of the algorithm.

```
void selectionSort2(int a[], int n)
{
    int min_id, max_id, front = 0, end = n-1;

    while ( __(5)__ ) {
        min_id=max_id=front;
        for (int i=front;i<=end;i++){
            min_id= __(6)__ ;
            max_id= __(7)__;
        }
        int t = a[max_id];
        a[max_id] = a[end];
        a[end] = t;
        if( __(8)__ ) min_id = max_id;
        t = a[min_id];
        a[min_id] = a[front];
        a[front] = t;
        front++;
        end--;
    }
    return;
}
```

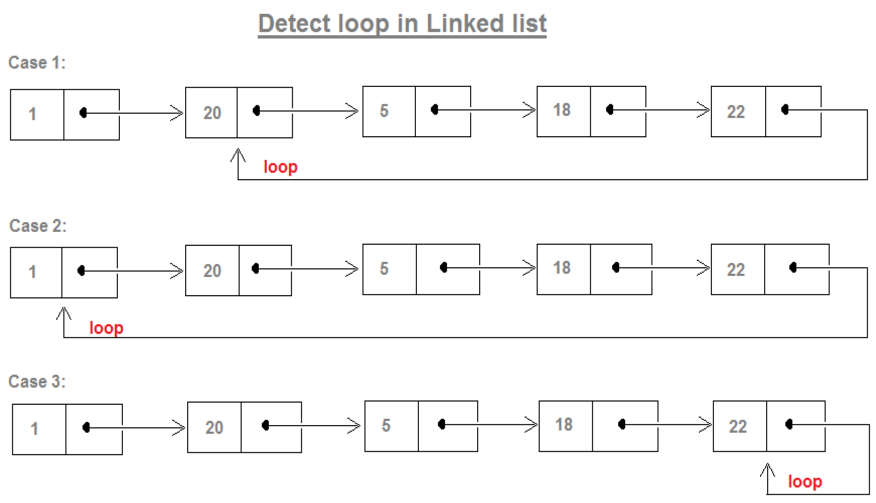
3. The following code is based on our course's libgraphics library to implement the cursor blinking(光标闪烁) in the center of the window. Please fill in the blanks to complete the program.

```
/*To reduce space, omit the relevant header file include code*/
```

```
#define TIMER_CURSOR  1
#define CURSOR    "_"

static double cursorx, cursory;
static bool isDisplayCursor = TRUE;
void TimerEventProcess(int timerID);

void Main()
{
    ___(9)___;
    registerTimerEvent(___(10)___);
    cursorx = GetWindowWidth()/2; cursory = GetWindowHeight()/2;
    MovePen(cursorx, cursory);
    DrawTextString(CURSOR);
   ___(11)___(TIMER_CURSOR, 500);/*start 500ms cuorsor blinking timer */
}

void TimerEventProcess(int timerID)
{
   bool erasemode;

   if (TIMER_CURSOR == ___(12)___) {
      erasemode = GetEraseMode();
      isDisplayCursor = ___(13)___;
      SetEraseMode(isDisplayCursor);
     ___(14)___;
      DrawTextString(CURSOR);
      SetEraseMode(___(15)___);
   }
}
```

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

1. Given the following code fragments, write a function "***LoopDetect***" to check if the linked list has loop or not. Below diagrams show linked lists with a loop.



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

struct Node {
   int data;
   struct Node* next;
};

void push(struct Node** head_ref, int new_data)
{
   /* allocate node */
   struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));

   /* put in the data */
   new_node->data = new_data;

   /* link the old list off the new node */
   new_node->next = (*head_ref);

   /* move the head to point to the new node */
   (*head_ref) = new_node;
}

int LoopDetect (struct Node* list)
{
   ……
}

int main()
{
   struct Node* head = NULL;

   push(&head, 20);
   push(&head, 4);
   push(&head, 15);
   push(&head, 10);
   head->next->next->next->next = head;

   if (LoopDetect (head))
            printf("Loop found");
   else
            printf("No Loop");
   return 0;
}
```

2. Write a function ***int RemoveDuplicates(int array[], int n)*** that removes all duplicate (重复的) value from a sorted array of integers, leaving only a single copy of each value and returning the effective size of the new array. The order of the original array elements should remain unchanged. Expected time complexity is ***O(n)*** and extra space is ***O(1)***.

   Example input:
      array: 65 72 75 79 82 82 84 84 84 86 90 94 95, n: 13

   Example result:
      array: 65 72 75 79 82 84 86 90 94 95, return value: 10

```
int RemoveDuplicates(int array[], int n)
{
    ……
}
```