計算機程式二 111 學年度第 2 學期小考 4 試題

系級：_____學號：_____ 姓名：_____

1. (1) __struct node_s__ (2) ____NULL____ (3) ____current____
(4) ____back____ (5) ____newnode____ (6) ____root____

請完成以下程式碼，使執行結果符合輸出。(Please complete the empty space to make the execution result match the output.)

The output of void f1(): 4 5 6 1 3 9 8 2

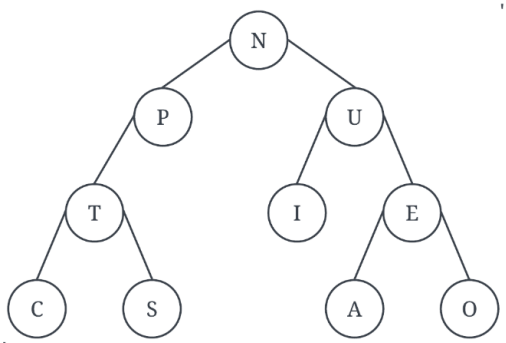
```
typedef struct node_s{
    int data;
    _____ *right, *left;                    // (1)
} tree_t;
typedef tree_t *btree;
btree create_btree(btree root, int val){
    btree newnode, current, back;
    newnode = (btree)malloc(sizeof(tree_t));
    newnode->data = val;
    newnode->left = newnode->right = NULL;
    if (root == _____ ){                    // (2)
        root = newnode;
        return root;}
    else{
        current = root;
        while (____ != NULL){               // (3)
            ____ = current;                 // (4)
            if (current->data > val) current = current->left;
            else current = current->right;
        }
        if (back->data > val) back->left =_____ ;   // (5)
        else back->right = newnode;
    }
    return ____ ;}                          // (6)
int f1(){
    int arr[] = {4, 5, 6, 1, 3, 9, 8, 2};
    btree ptr; int val, i; ptr = NULL;
    for (i = 0; i < 8; i++){
        ptr = create_btree(ptr, arr[i]);
        printf("%d ", arr[i]);
    }}
```

2. 根據中序與後序的二元樹走訪方式，請畫出唯一的二元樹 (According to the inorder and the postorder binary tree walk, please draw the unique binary tree)

中序(Inorder): CTSPNIUAEO

後序(Postorder): CSTPIAOEUN

Ans:



3. (1) __root->left__ (2) __root->right__ (3) __root->left__
(4) __root->right__ (5) __root->left__ (6) __root->right__

請完成以下程式碼，使執行結果符合輸出。(Please complete the empty space to make the execution result match the output.)

The output of void f3():
6 3 2 1 4 5 9 7
1 2 3 4 5 6 7 9
1 2 5 4 3 7 9 6

```
void preorder(btree root) {
    if ( root != NULL ) {
        printf("%d ",root->data);
        preorder( _____ );                  // (1)
        preorder( _____ );                  // (2)
    }}
void inorder(btree root) {
    if (root!=NULL) {
        inorder( _____ );                   // (3)
        printf("%d ", root->data);
        inorder( _____ );                   // (4)
    }}
void postorder(btree root) {
    if (root!=NULL) {
        postorder( _____ );                 // (5)
        postorder( _____ );                 // (6)
        printf("%d ",root->data);
    }}
int f3(){
    int arr[] = {6, 3, 2, 1, 4, 9, 7, 5};
    btree ptr; int val, i; ptr = NULL;
    for (i = 0; i < 8; i++){
        ptr = create_btree(ptr, arr[i]);}
    preorder(ptr);
    printf("\n");
    inorder(ptr);
    printf("\n");
    postorder(ptr);
}
```

4.說明一個 ADT(Abstract Data Type)的優點，並舉出兩個 ADT 的例子(Explain an advantage of ADT (Abstract Data Type), and give two examples of ADT)

Ans:

優點:

1.客戶端程式呼叫存取,僅需操作其所提供的介面函式。

2.不用也不需要知道內部如何實作。

3.系統具高維護性與彈性,不論擴充或重構,客戶端程式僅受最小幅度影響。

例子:

1. stack 2. queue 3. circular queue 4. set

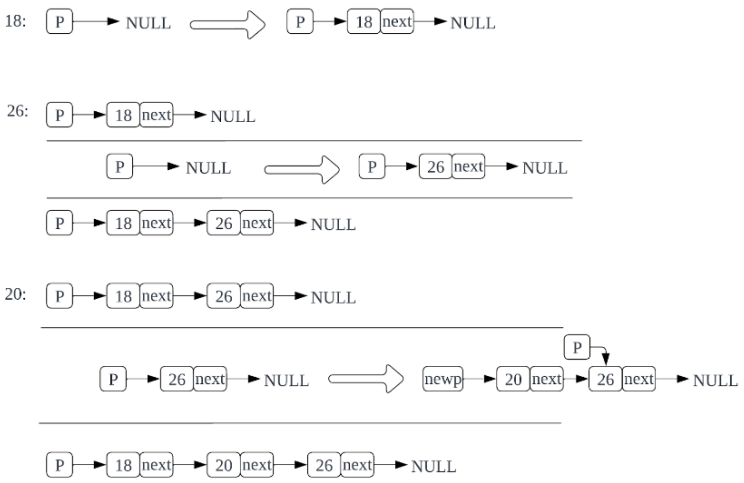
5. map 6.Priority Queue 7.Graph 8. tree

5. LinkList 一開始為空，依序加入資料:18, 26, 20，請以以下函式畫出節點加入過程(LinkList is empty at the beginning, add data in order: 18, 26, 20, please use the following function to draw the node adding process) :

Function:

```
nodep_t insertInOrderR(nodep_t p, int data) {
 nodep_t newp, current, prev;
 if (p ==NULL) { return create(data); }
 else if ((p->data) >= data) {
      newp = create(data);
      newp->next = p;
      return newp; }
 else {
      p->next = insertInOrderR(p->next, data);
      return p; }}
```

Ans:



6. Extern 是什麼? 以下關於 Extern 使用範例是否有誤，若有錯誤，該如何修正? (What is Extern? Are there any errors in the following examples of Extern usage, and if so, how should we fix them?)

```
// example.c
#include <stdio.h>
int main() {
extern double some_var = 100;
printf("%f\n", some_var);
return 0;
}
```

Ans:

1. 使用外部檔案定義的變數，外部檔案是指下 Extern 這個指令的檔案之外的檔案。

2. 有誤，修正方式如下:

```
extern double some_var;
some_var = 100;
```

7. Header file 和 C source file 分別包含什麼內容?(What does the Header file and C source file contain respectively?)

Ans:

Header file: 註解區塊,說明函式庫的功能目的、#define 宣告、命名、常數和巨集、型別定義 typedef

C source file: 用來撰寫程式邏輯內容

8.加入哪些指令於自訂 header 檔案 testH.h 中,使得無論在程式中的 directives 指令 #include " testH.h" 多少次, testH.h 都只會被編譯一次。請使用正確命名法則? (What commands should be added to the custom header file “testH.h” so that no matter how many times the commands #include "testH.h" in the program, “testH.h” will only be compiled once. Please use the correct naming convention.)

Ans:
```
#if !defined (TESTH_H_INCL)
#define TESTH _H_INCL
…
#endif
```

9. (1) __stack_node_s___ (2) ___*topp____ (3) ___sp->topp___ (4) _p->restp_ (5) __p__ (6) _newNode->restp_ (7) _sp->topp_

請完成以下程式碼，使執行結果符合輸出。(Please complete the empty space to make the execution result match the output.)

The output of void f9(): 6 + 3 E

```
#define ERROR 'E'
typedef char stack_element_t;
typedef struct stack_node_s {
    stack_element_t element;
    struct _____ *restp;                    // (1)
} stack_node_t;
typedef struct {
    stack_node_t ___;                       // (2)
} stack_t;
stack_element_t pop(stack_t *sp) {
    stack_element_t data;
    stack_node_t *p = ____ ;                // (3)
    if (p == NULL)
        return ERROR;
    data = p->element;
    sp->topp =____ ;                        // (4)
    free(__);                               // (5)
    return data;
}
void push(stack_t *sp, stack_element_t data){
    stack_node_t *newNode;
    newNode = (stack_node_t *)
        malloc(sizeof(stack_node_t));
    newNode->element = data;
    ____ = sp->topp;                        // (6)
    ____ = newNode;                         // (7)
}
int f9(){
    stack_t s = {NULL};
    push(&s, '3'); push(&s, '+'); push(&s, '6');
    printf("%c ", pop(&s));
    printf("%c ", pop(&s));
    printf("%c ", pop(&s));
    printf("%c ", pop(&s));
    return 0;
}
```

10. (1) qp->frontp_ (2) ERROR_ (3) _NULL_ (4) p->element_ (5) __newNode__ (6) __return__ (7) __qp->rearp->restp__

請完成以下程式碼，使執行結果符合輸出。(Please complete the empty space to make the execution result match the output.)

The output of void f10(): 1 E + 2

```
#define ERROR 'E'
typedef char queue_element_t;
typedef struct queue_node_s {
    queue_element_t element;
    struct queue_node_s *restp;
} queue_node_t;
typedef struct {
    queue_node_t *rearp;
    queue_node_t *frontp;
} queue_t;
queue_element_t dequeue(queue_t *qp) {
    queue_element_t data;
    queue_node_t *p = ____ ;                // (1)
    if (p == NULL)
        return _____ ;                      // (2)
    else if (p == qp->rearp)
        qp->rearp = ____ ;                  // (3)
    data =____ ;                            // (4)
    qp->frontp = p->restp;
    free(p);
    return data;
}
void enqueue(queue_t *qp, queue_element_t data){
    queue_node_t *newNode;
    newNode = (queue_node_t *)
        malloc(sizeof(queue_node_t));
    newNode->element = data;
    newNode->restp = NULL;
    if (qp->rearp == NULL)
    {
        qp->frontp = newNode;
        qp->rearp = ____ ;                  // (5)
        ___ ;                               // (6)
    }
    _______ = newNode;                      // (7)
    qp->rearp = newNode;
}
int f10(){
    queue_t s = {NULL, NULL};
    enqueue(&s, '1');
    printf("%c ", dequeue(&s));
    printf("%c ", dequeue(&s));
    enqueue(&s, '+');
    enqueue(&s, '2');
    printf("%c ", dequeue(&s));
    printf("%c ", dequeue(&s));
    return 0;
}
```

11. Class 分為 abstract 和 concrete，分別說明何為 abstract class 和 concrete class ，並且說明什麼是 inheritance？(Class is divided into abstract and concrete, explain what is abstract class and concrete class respectively, and explain what is inheritance?)

Ans:

1. abstract class :一種無法實例化的類別，只能被繼承。常用於定義共用的屬性和方法，並包含未實作的抽象方法。

2. concrete class : 可以實例化的類別，可以直接創建物件，並實作繼承而來的抽象方法。

3. inheritance : 繼承是允許一個類別繼承另一個類別的屬性和方法的機制。通過繼承，可以建立物件間的階層結構，讓程式碼更具結構性、可重複使用性和可擴展性。