Exam A&DinC, with answers Friday, April 10, 2015, 9 - 12 h.

Your personal information (10 points)

Name:	
Student Number:	
Study Program:	

Exam instructions
– please read carefully –

- Please write your name, student number, and study program on this page.
- The first part of the exam consists of 13 multiple choice questions. Read carefully each question and the four options, and select the option that answers the question correctly. When more than one option answers the question correctly, select the most informative option.
  Your should provide your answers for the multiple choice questions **on a separate multiple choice form**. Do not forget to write your name and student number on that form.
- The second part of the exam consists of 3 open problems. Write your answers in the boxes below the problems. If you need additional space, use the last page of the exam, and provide a reference to the problem.
- Your exam grade is computed as follows. You obtain 10 points for filling in your personal information on this page. For n correct answers to the multiple choice questions, you will earn $\mathsf{max}(3n - 9, 0)$ points. So the maximal number of points for the multiple choice questions is 30, and any wrong or missing answer will cost you 3 points. For each of the 3 open problems you can earn maximally 20 points. Your exam grade is $p/10$ where p is the number of points you earned.

This is a version of the exam with answers. The correct answers to the multiple choice questions are printed in boldface.

Part I: Multiple Choice Questions

1. Consider the following definition of the type `Stack`:

```
typedef struct Stack {
  int *array;
  int top;
  int size;
} Stack;
```

The `top` of a `Stack` is the index of the first free position in `array`, so e.g. in an empty stack we have `top==0`. The function `push` is defined by

```
1  void push (int value, Stack *stp) {
2    if (stp->top == stp->size) {
3      doubleStackSize(stp);
4    }
5    stp->top++;
6    stp->array[stp->top] = value;
7  }
```

Which of the following is correct?

A. This is a correct definition of `push`.

B. When lines 2-4 are moved to the end of the function definition, this is a correct definition of `push`.

C. When lines 5 and 6 are interchanged, this is a correct definition of `push`.

D. When lines 2-4 are moved to the end of the function definition and lines 5 and 6 are interchanged, this is a correct definition of `push`.

2. The function `findInList()` is defined by

```
1  int findInListIt(List li, int n) {
2    while ( ??? ) {
3      li = li->next;
4    }
5    if ( li == NULL ) {
6      return 0;
7    } else {
8      return 1;
9    }
10 }
```

What should replace the `???` part in line 2 in order to obtain a correct function?

A. `li != NULL && li->item != n`

B. `li != NULL || li->item != n`

C. `li->item != n && li != NULL`

D. `li->item != n || li != NULL`

3. Consider the following grammar:

   ⟨*code*⟩ ::= ⟨*word*⟩ ⟨*number*⟩ .

   ⟨*number*⟩ ::= ⟨*digit*⟩ { ⟨*digit*⟩ } .

   ⟨*word*⟩ ::= [⟨*capital*⟩] { ⟨*lowercase*⟩ } .

   ⟨*capital*⟩ ::= 'A' | 'B | 'C' .

   ⟨*lowercase*⟩ ::= 'a' | 'b' | 'c' .

   ⟨*digit*⟩ ::= '0' | '1' | '2' | '3' .

   Which string is **not** a production of ⟨*code*⟩?

   A. ab13
   B. ABc0311
   C. 123
   D. Aaa2

4. Let G be an ambiguous grammar. Which of the following statements is correct?

   A. For some strings, it is not certain whether they can be produced by G.
   B. Some strings can be produced by G in more than one way.
   C. Recognizing whether a string is produced by G is problematic, since G is ambiguous.
   D. The evaluation of strings produced by G is possible thanks to the ambiguity of G.

5. We consider the array representation of binary trees where the root has index 1. Which of the following statements is correct?

   A. The children of a node with index n have index $2n + 1$ and $2n + 2$.
   B. The parent of a node with index $n > 1$ has index n div 2.
   C. If $m < k < n$ and there are nodes with indices m and n, then there is a node with index k.
   D. For every k less that the height of the tree, there is a node with index 2^k.

6. Let T be a binary search tree containing integers, with n nodes and height h. Which of the following statements is **wrong**?

   A. Inorder traversal of T yields the integers in ascending order.
   B. Deciding whether value x occurs in T can be done in $\mathcal{O}(h)$ time.
   C. If node k contains x and node l is a right descendant of k that contains y, then $x < y$.
   D. h is in $\mathcal{O}(\log(n))$.

7. The function `recognizeExpression` is defined by

```
void recognizeExpression() {
  char *ar = readInput();
  List tl = tokenList(ar);
  if ( acceptExpression(&tl) ) {
    printf("this␣is␣an␣expression\n");
  } else {
    printf("this␣is␣not␣an␣expression\n");
  }
  free(ar);
  freeTokenList(tl);
}
```

So it reads an input string, transforms it into a token list, and checks whether the token list represents an expression. It uses the function with prototype `int acceptExpression(List *lp);` that scans through the token list referred to by `lp` and indicates whether it has observed an expression.

A. This is a correct definition.

B. This is not a correct definition: it will accept strings that are not expressions (e.g. `42))*(+ `).

C. This is not a correct definition: it will lead to memory leaks.

D. This is not a correct definition: it will accept strings that are not expressions (e.g. `42))*(+ `), and it will lead to memory leaks.

8. We use the array representation of binary trees where the root has index 1. Consider the following definition of the function `upheap`:

```
1 void upheap (Heap *hp, int n) {
2   if ( ??? && hp->array[n] > hp->array[n/2] ) {
3     swap(&(hp->array[n]),&(hp->array[n/2]));
4     upheap(hp,n/2);
5   }
6 }
```

It uses the function `swap` that interchanges the values referred to by its arguments. What should replace the `???` part in line 2 in order to obtain a correct function?

A. `n!=0`

B. `n>0`

C. `n>1`

D. `n>2`

9. Which statement about the algorithm Heapsort is correct?

   A. Its time complexity is $\mathcal{O}(n)$.

   B. Its time complexity is $\mathcal{O}(n \log(n))$.

   C. Its time complexity is $\mathcal{O}(n\sqrt{n})$.

   D. Its time complexity is $\mathcal{O}(n^2)$.

10. Let a collection W of $m > 1$ words be given, with n the sum of the lengths of the words. Let ST be a standard trie for W, and CT a compressed trie (i.e. a trie where every node except the root contains a nonempty string). Which of the following statements is **wrong**?

    A. CT contains no nodes with branching degree 1, while ST may contain such nodes.

    B. The number of nodes in ST is in $\mathcal{O}(n)$, while CT has at most $2m$ nodes.

    C. The memory required for ST is in $\mathcal{O}(n)$, while for CT it is in $\mathcal{O}(m)$.

    D. ST and CT have the same number of leaves.

11. Let T be a text with length n, and let ST be the suffix trie for text T. Moreover, let P be a pattern with length m. Which of the following statements is **wrong**?

    A. ST has at most $2n$ nodes.

    B. The memory required for ST is in $\mathcal{O}(n)$.

    C. P occurs in T if and only if there is a path in ST from the root to a leaf that corresponds with P.

    D. ST enables checking whether P occurs in T in $\mathcal{O}(m)$ time.

12. Which of the following statements about connected graphs is correct?

    A. Every two nodes are connected by a path.

    B. There is an edge between every pair of different nodes.

    C. (number of edges) $\geq$ (number of nodes).

    D. When a graph is not connected, there may be edges that are not connected to any node.

13. Which of the following statements about graphs is correct?

    A. The number of nodes with even degree is even.

    B. The number of nodes with even degree is odd.

    C. The number of nodes with odd degree is even.

    D. The number of nodes with odd degree is odd.

Part II: Open Questions

1. [20 points] The type `List` is defined by

```
typedef struct ListNode *List;

struct ListNode {
  int item;
  List next;
};
```

Define a C function with prototype `List removeItemAndNext(List li, int n);` that removes the first occurrence of value `n` in list `li` and also removes the next item in the list. When `n` does not occur in the list, nothing happens. When `n` only occurs at the end of the list, only that occurrence is removed. Do not forget to free memory whenever required, so that no memory leaks arise.

Solution:

```
/**** recursively ****/
List removeItemAndNextRec(List li, int n) {
  List returnList = NULL;
  if ( li == NULL ) {
    return li;
  }
  if ( li->item == n ) {
    if ( li->next != NULL ) {
      returnList = (li->next)->next;
      free(li->next);
    }
    free(li);
    return returnList;
  }
  li->next = removeItemAndNextRec(li->next,n);
  return li;
}
```

```
/**** with an auxiliary function ****/

List removeFirstNode(List li) { /* precondition: li != NULL */
  List returnList = li->next;
  free(li);
  return returnList;
}

List removeItemAndNext(List li, int n) {
  if ( li == NULL ) {
    return li;
  }
  if ( li->item == n ) {
    li = removeFirstNode(li);
    if ( li == NULL ) {
      return li;
    }
    return removeFirstNode(li);
  }
  li->next = removeItemAndNext(li->next,n);
  return li;
}

/**** iteratively ***/
List removeItemAndNext(List li, int n) {
  List prev = NULL;
  List loc = li;
  while ( loc != NULL && loc->item != n ) {
    prev = loc;
    loc = loc->next;
  }
  if ( loc != NULL ) {  /* so loc->item == n */
    if ( prev != NULL ) {
      prev->next = loc->next;
    } else {
      li = loc->next;
    }
    if ( loc->next != NULL ) {
      if ( prev != NULL ) {
        prev->next = loc->next->next;
      } else {
        li = loc->next->next;
      }
      free(loc->next);
    }
    free(loc);
  }
  return li;
}
```

2. [20 points] The type `Tree` is defined by

```
typedef struct TreeNode *Tree;

struct TreeNode {
  int item;
  Tree leftChild, rightChild;
};
```

Define a C function with prototype `Tree mirrorCopy(Tree t);` that makes a mirror copy of its argument. This is a copy in which for every node the left child and the right child are interchanged. Your function should leave the argument unchanged.

Solution:

```
Tree mirrorCopy(Tree t) {
  Tree tNew = NULL;
  if ( t != NULL ) {
    tNew = malloc(sizeof(struct TreeNode));
    assert(tNew != NULL);
    tNew->item = t->item;
    tNew->leftChild = mirrorCopy(t->rightChild);
    tNew->rightChild = mirrorCopy(t->leftChild);
  }
  return tNew;
}
```

3. [20 points] Define in pseudocode an algorithm CycleCheck that checks whether a connected simple graph contains a proper cycle (i.e. a cycle in which no two edges are equal). *Hint*: consider a variant of Depth-First Search.

Solution:

```
algorithm CycleCheck(G)
    input connected simple graph G
    output YES if G contains a proper cycle, otherwise NO
    let v be a node of G
    return CycleCheck(G,v)

algorithm CycleCheck(G,v)
    input connected simple graph G with node v
    output YES if G contains a proper cycle, otherwise NO
    give v the label VISITED
    forall unlabeled e incident with v do
        w ← the other node incident with e
        if w has label VISITED then /* we have a cycle! */
            return YES
        /* new node discovered */
        give e the label NEW
        if CycleCheck(G,w) then
            return YES
    return NO
```