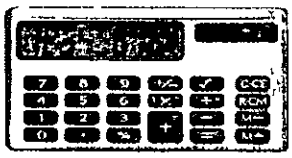
UNIVERSITY EXAMINATIONS



UNIVERSITEITSEKSAMENS

UNISA university of south africa

COS2611

October/November 2016

PROGRAMMING: DATA STRUCTURES

Duration 2 Hours

70 Marks

EXAMINERS

FIRST MR KT MASOMBUKA
SECOND MR CL PILKINGTON

Use of a non-programmable pocket calculator is permissible

Closed book examination

This examination question paper remains the property of the University of South Africa and may not be removed from the examination venue

This paper consists of 7 pages + Appendix A, B, C and D (pp 8 - 9)

INSTRUCTIONS

1. Answer all questions
2. All rough work must be done in your answer book
3. The mark for each question is indicated in brackets next to the question

GOOD LUCK

QUESTION 1 [20]

For each of following questions choose the correct alternative

1 1 What is the Big-Oh value for the following function $3n^2 + n \log n + 6n^3$?

A $O(N^2)$
B $O(N \log N)$
C $O(N^3)$
D $O(\log N)$
E $O(N^5)$

1 2 What is the running time of the following code fragment?

```
int sum = 0,
int i = 2,
while (i < n)
{
    sum += i,
    i*=2,
}
```

A O(1)
B O(N)
C O(log N)
D O(N log N)
E $O(N^2)$

Questions 1 3, 1 4 and 1 5 refer to the following code fragment

```
1 for(int i = 0, i < n; i++ )
2         sum--,
3   for(int j = 1, j <= n, j*=2 )
4         for(int k = 1, k <= n, k++ )
5             sum++,

6 for(int p = n, p > 0; p/=2 )
7    for(int q = 0; q < n, q++ )
8         sum--,
```

1 3 How many times is statement 5 executed?

A O(N)
B $O(N^2)$
C $O(N^3)$
D O(N log N)
E O(log N)

1 4 How many times is statement 8 executed?

A O(N)
B $O(N^2)$
C $O(N^3)$
D O(log N)
E O(N log N)

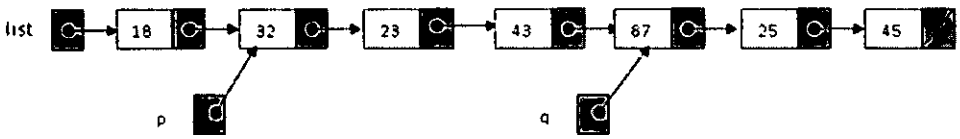
1 5 What is the running time of the entire code fragment?

A O(N)
B $O(N^2)$
C $O(N^3)$
D O(log N)
E O(N log N)

1 6 An algorithm takes 10 seconds for an input of size 10 If the running time of the algorithm is O(log N), approximately how long does it take to solve a problem of size 100?

A 20s
B 10s
C 100s
D 1000s
E None of the above

Consider the following list where each node is of type `nodeType` (see appendix A) and answer question 1 7, 1 8 and 1 9.



1 7 What is the output of the following C++ code?

```
while (p!=NULL){
   cout << q->info << ", ",
   p = p -> link,
   q = p,
}
```

A 87, 23, 43, 87, 25, 45,
B 87, 25, 45, 32, 23, 43,
C 87, 32, 23, 25, 45,
D 87,25,45,

E None of the above

1 8 What is the output of the following C++ code?

```
while (q!=NULL){
   q = q->link,
   p = p -> link ->link,
   cout << p->info << ", ";
}
```

A 32, 23, 43,
B 23, 43, 87 ,25,
C 23, 43, 87,
D 43, 25,
E None of the above

1 9 What is the output of the following C++ code?

```
while (q!=NULL){
   p = p -> link,
   cout << q->info << ", ",
   q = p ->link-link,
}
```

A 32, 23, 43
B 23, 43, 87,25
C 87, 25, 45
D 87, 87, 25, 45
E None of the above

1 10 A linked list is not a random access data structure such as a(n) ____

A. Stack
B Pointer
C Queue
D Array
E None of the above

QUESTION 2 [10]

Consider the definition of `class linkedListType` given in appendix A Provide a member function of `class linkedListType` that deletes the minimum element in a list Use the following header

```
template <class Type>
void linkedListType<Type>  removeMin(),
```

Hint declare the following variables

```
nodeType<Type> *current = first ->link,
nodeType<Type> *prevCurrent = first;
nodeType<Type> *smallest = first,
nodeType<Type> *prevSmallest = first,
```

`current` and `prevCurrent` should be used to traverse the list, while `smallest` should point to the smallest node so far and `prevSmallest` points to the node preceding `smallest`

You can assume that the list is not empty and that operators > and < have been overloaded for `Type`

Note The function `deleteNode()` is not a member of class `linkedListType`

QUESTION 3 [6]

Consider the sorted array `sArray` of integers below

4	8	19	25	34	39	45	48	66	75	89	95

The Binary Search algorithm uses two index variables, first and last to determine the position (stored in another index variable mid) to search for an element Using the Binary Search algorithm to search for the element **89** in `sArray`, give the values of `sArray[first]`, `sArray[mid]` and `sArray[last]` (in the form of a table) every time a new value is calculated for mid

QUESTION 4 [8]

4 1 State whether a **Queue**, a **Stack** or **Neither** would be appropriate for each of the following tasks Indicate why or why not

i A list of customers waiting to be seated in a restaurant (½)

ii An address book listing names and telephone numbers in alphabetical order (½)

iii A word processor must allow a line of characters to be edited Pressing the backspace key deletes the previous character and pressing CTRL and backspace deletes the entire line Users must be able to undo deletion operations (½)

iv A programming team accepts jobs and prioritises them on the basis of urgency (½)

4 2 Write a function template, `identicalQ` that checks whether two queues provided as parameters are identical If that is the case, the function should return the Boolean value true or false otherwise Use the following header

```
template <class Type>
bool identicalQ (queueType <Type> Q1, queueType <Type> Q2);
```

You may use any member function of class `queueType` given in appendix C (6)

QUESTION 5 [8]

$L=\{a^{n+1}b^n\}$ where $n \geq 1$, is a language of all words with the following properties

- The words consist of strings of a's followed by b's
- The number of a's is **equal** to the number of b's plus **1** (There is one extra a)
- Examples of words that belong to L are

  aab, where n=1 ,
  aaabb, where n=2 ,
  aaaabbb, where n=3 ,
  aaaaabbbb, where n=4

One way to test if a word w belong to this language is to use a stack to check if the number of a's balances the number of b's Use the provided header and write a function `isInLanguageL` that uses a local stack to test if any word belongs to L

```
bool isInLanguageL (string w),
```

You can make use of any member functions of `class stackType` (See appendix B) Note that this function is **not** a member of the `class stackType`

QUESTION 6 [5]

Consider the following sequence of numbers 10, 30, 40, 50, 20

Sort the list using Quick sort algorithm with the middle element as pivot Show the state of the list after each call to the partition procedure **Indicate the pivots in each list**

QUESTION 7 [5]

Write a recursive function `treeHeight` that calculates the height of a binary search tree (See appendix D) The height of a binary search tree is the maximum level in the tree Use the following driver function and header

Driver function

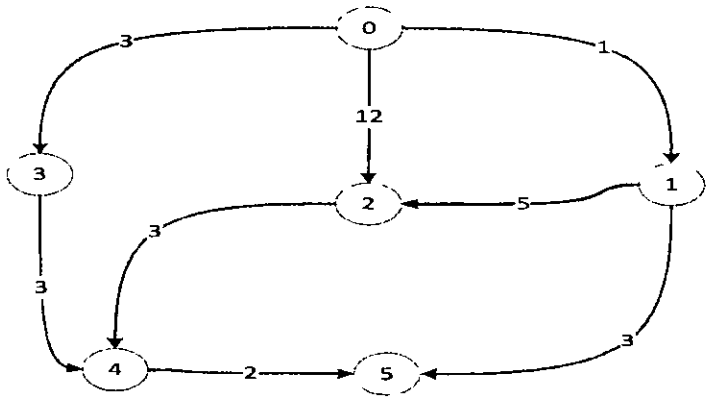
```
template<class Type>
int bSearchTreeType<Type>  treeHeight( )
{
     if (root != NULL)
          return treeHeight( root );
     return -1,
}
```

Header

```
template<class Type>
int bSearchTreeType<Type>: treeHeight(bSearchTreeType<Type> *p)
```

QUESTION 8 [8]

Consider the graph below and answer the following question



Use the shortest path algorithm to find the shortest distance from node 0 to every other node of the graph Give only the contents of the arrays `smallestWeight` and `weightFound` for each iteration of the algorithm Do not redraw the diagram

Appendix A

```
template <class Type>
struct nodeType
{
      Type info,
      nodeType<Type> *link,
},


template <class Type>
class linkedListType
{
public
    const linkedListType<Type>& operator=
                        (const linkedListType<Type>&),
    void initializeList(),
    bool isEmptyList() const,
    void print() const,
    int length() const,
    void destroyList(),
    Type front() const,
    Type back() const,
    bool search(const Type& searchItem),
    void insertFirst(const Type& newItem),
    void insertLast(const Type& newItem),
    linkedListIterator<Type> begin();
    linkedListIterator<Type> end(),
    linkedListType(),
    linkedListType(const linkedListType<Type>& otherList),
   ~linkedListType(),

    void removeMin();

protected
    int count;
    nodeType<Type> *first,
    nodeType<Type> *last,
private
    void copyList(const linkedListType<Type>& otherList),
},
```

Appendix B

```
template <class Type>
class stackType
{
public:
     void initializeStack(),
     bool isEmptyStack() const,
     bool isFullStack() const,
     void push(const Type& newItem),
     Type top() const,
     void pop(),
};
```

Appendix C

```
template <class Type>
class queueType
{
public
     bool isEmptyQueue() const,
     bool isFullQueue() const,
     void initializeQueue(),
     Type front() const,
     Type back() const,
     void addQueue(const Type& queueElement);
     void deleteQueue(),
private
     int count,
},
```

Appendix D

```
template <class Type>
class bSearchTreeType  public binaryTreeType<Type>
{
public
    bool search(const Type& searchItem) const,
    void insert(const Type& insertItem),
    void deleteNode(const Type & deleteItem),
    int treeHeight( );
private
    void deleteFromTree(binaryTreeNode<Type>* &p);
},
```

©
UNISA 2016