Model Answer

Benha University
1st Term (2013/2014) Final Exam
Class: 2nd Year Students
Subject: Data Structures

Faculty of Computers & Informatics
Date: 26/12/2013
Time: 3 hours
Examiner: Dr. Essam Halim

Answer the following questions:

Section I You should attempt TWO out of section I (questions 1, 2, 3 and 4) [Total 20 MARK]

Q1: A **heap** can be used to sort a set of elements, **illustrate the steps to sort** the following list of (33, 42, 67, 23, 44, 49, and 74), using heap, and **Write a C++ function** that to implement the algorithm of creating heap H? **(10 Marks)**

CREATING A HEAP ALGORITHM

Let H be a heap with n elements stored in the array HA. This procedure will insert a new element data in H. LOC is the present location of the newly added node. And PAR denotes the location of the parent of the newly added node.

1. Input first element in the array HA
2. Add new element, increment the size of HA, n=n + 1 and LOC = n
3. Repeat step 4 to 7 until (LOC <= 1)
4. PAR = LOC/2
5. If (data <= HA[PAR])
   (a) HA[LOC] = data
   (b) Exit
6. HA[LOC] = HA[PAR]
7. LOC = PAR
8. HA[LOC] = data
9. Exit

```
//Creating Heap Tree
#include <iostream.h>
#include <conio.h>
class heap
{
int k[11],size;
public:
void getdata(void);
friend void create_heap(heap &);
void showdata(void);
};
void heap :: getdata(void)
{
clrscr();
cout<<"Enter the size of Array :-";
cin>>size;
cout<<"\nEnter "<<size<<" Elements\n";
 for(int i=1;i<=size;i++) //Creating heap from  index1 instead of index0
   cin>>k[i];
}
void heap :: showdata(void)
{
  clrscr();
  cout<<"\n\nHeap Function Output\n\n";
  for(int i=1;i<=size;i++)
    cout<<k[i]<<endl;
}
void create_heap(heap &a)
{
int q,i,j,key;
  for(q=2;q<=a.size;q++)
  {
    i=q;
    key=a.k[i];
    j=i/2;
    while(i>1 && key>a.k[j])
      {
         a.k[i]=a.k[j];
         i=j;
         j=i/2;
         if(j<1)
            j=1;
      }
    a.k[i]=key;
    }
}
void main()
```

```
{
heap o1;
o1.getdata();
create_heap(o1);
o1.showdata();
getch();
}
```

Q2: Write an algorithm to **merge two sorted arrays** into a third array? **(10 Marks)**

Ans Q4:

```
void merge(int low, int mid, int high) {
   int h, i, j, b[50], k;
   h = low;
   i = low;
   j = mid + 1;
   while ((h <= mid) && (j <= high)) {
      if (a[h] <= a[j]) {
         b[i] = a[h];
         h++;
      } else {
         b[i] = a[j];
         j++;
      }
      i++;
   }
   if (h > mid) {
      for (k = j; k <= high; k++) {
         b[i] = a[k];
         i++;
      }
   } else {
      for (k = h; k <= mid; k++) {
         b[i] = a[k];
         i++;
      }
   }
   for (k = low; k <= high; k++)
      a[k] = b[k];
}
```

Q3: Write a C++ function to **reverse** a **singly linked list**? **(10 Marks)**

Ans Q5:

A procedure to reverse a singly linked list:

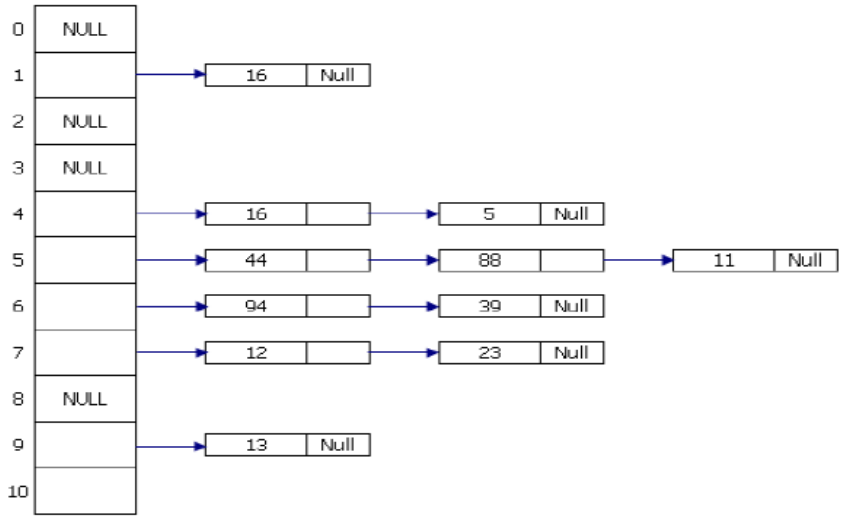
```
reverse(struct node **st)
{
struct node *p, *q, *r;
p = *st;
q = NULL;
while(p != NULL)
{
r =q;
q = p;
p = p _ link;
q _ link = r;
}
*st = q;

}
```

Q4: Draw the 11 item **hash table** resulting from hashing the keys: 12, 44, 13, 88, 23, 94, 11, 39, 20, 16 and 5 using the hash function **h(i) = (2i+5) mod 11**? **(10 Marks)**

Ans.

table size is 11

h(12) = (24+5) mod 11 = 29 mod 11 = 7
h(44) = (88+5) mod 11 = 93 mod 11 = 5
h(13) = (26+5) mod 11 = 31 mod 11 = 9
h(88) = (176+5)mod 11= 181mod 11 = 5
h(23) = (46+5) mod 11 = 51 mod 11 = 7
h(94) = (188+5)mod 11= 193mod 11 = 6
h(11) = (22+5) mod 11 = 27 mod 11 = 5
h(39) = (78+5) mod 11 = 83 mod 11 = 6
h(20) = (40+5) mod 11 = 45 mod 11 = 1
h(16) = (24+5) mod 11 = 29 mod 11 = 4
h(5) = (10+5) mod 11 = 15 mod 11 = 4



Section II You should attempt ALL of section II (Questions 5, and 6) [Total 40 MARK]

Q5: outlines what is wrong in the following codes **and rewrite the wrong and it's right ONLY**? **(20 Marks)**

a) Code 1

```
1-  int linklist::search(int num)   {
2-      node *temp;
3-      temp = p;  int count = 1;
4-     while (temp->link != NULL)
5-     {
6-        if (temp->data == num)
7-        {
8-           cout << "Data is :"<< temp->data<<
"\nPosition is : "<< count << endl;
9-           return count;
10-       } else        {
11-       temp = temp->link;
12-       count ++;    }
13-    }
13-    cout << "Data "<< num << "  Not found";  }
```

b) Code 2

```
1- void selectionSort (int arr[ ], int n)
2-  {
3-      int i, j, loc, swap;
4-      for (i = 0; i < n - 1; i++) {
5-          loc = i;
6-          for (j = i + 1; j < n; j++)
7-              if (arr[j] < arr[loc])
8-                  loc = i;
9-          if (loc != i) {
10-             swap = arr[i];
11-             arr[i] = arr[loc];
12-             arr[loc] = swap;
13-         }
14-     }
15- }
```

c) Code 3

```
1- void bubblesort (int *a , int n)
2- {
3-          int i , j, k, swap;
4-          for(i = 0 ; i < n ; i++)
5-          for(j = 0 ; j < n- i -1 ; j++)
6-                    if (a[ j ] > a[ j+1 ])
7-                    {
8-                              swap   = a[ j ];
9-                              a[ j ]    = a[ j+1 ];
10-                             a[ j+1 ] = swap;
11-                   }
12- }
```

d) Code 4

```
1- int linearSearch(int arr[], int data, int size)
2- {
3-   int i;
4-   for (i=0; i<size; ++i)
5-   {
6-     if (arr[i]== data)
7-     {
8-       return i;
9-     }
10-  }
11-  return -1; // not matching
12- }
```

Q6: Outlines what is wrong in the following algorithms **and rewrite the wrong and it's right ONLY**? **(20 Marks)**

A)- Inserting a node in DOUBLY LIST algorithm

1. Input the DATA and POS
2. Initialize Temp = START; i = 0
3. Repeat step 4 if (i < POS and TEMP !=NULL)
4. TEMP = TEMP → RPoint; i = i +1
5. If (TEMP != NULL) and (i == POS)
   - (a) Create a NNode
   - (b) NNode → DATA = DATA
   - (c) NNode → RPoint = TEMP → RPoint
   - (d) NNode → LPoint = Temp
   - (e) TEMP → RPoint) → LPoint = NNode
   - (f) TEMP → RPoint = NNode
6. Else
   - (a) Display "Position NOT found"
7. Exit

B)- Deleting the ROOT of a heap algorithm

1. Input n elements in the heap HA
2. Data = HA[1]; last = HA[n] and n = n – 1
3. LOC = 1, left = 2 and right = 3
4. Repeat the steps 5, and 6 while (right <= n)
5. If (HA[right] <= HA[left])
   - (i) HA[LOC] = HA[left]
   - (ii) LOC = left

   (b) Else
   - (i) HA[LOC] = HA[right]
   - (ii) LOC = right

6. left = 2 × LOC; right = left +1
7. HA[LOC] = last
8. Exit

C)- Binary search algorithm

1. Input an array A of n elements and DATA.
2. L = 0, U = n-1; mid = int ((L+U)/2)
3. Repeat 4:6 while (L <= U) & (A[mid] ! = DATA)
4. If (DATA < A[mid]), (a) U = mid–1
5. Else, (a) L = mid + 1
6. mid = int ((L + U)/2)
7. If (A[mid]== DATA)
   - (a) Display "the data found at mid"
8. Else
   - (a) Display "the data is not found"
9. Exit

D)- Insertion sort algorithm

1. Input an array A of n numbers
2. Initialize i = 1 and repeat through steps 4.
   - (a) If (i < n)
   - (b) SWAP = A [i],
   - (c) POS = i - 1
3. Repeat 3 if (SWAP < A[POS] and (POS >= 0))
   - (a) A [POS +1] = A [pos]
   - (b) POS = POS -1
4. A [POS +1] = SWAP
5. Exit

Section III You should attempt TWO out of section III (questions 7, 8, and 9) [Total 40 MARKS]

Q7: QUEUE is a data structure, which uses First in First out (FIFO) principle. **Write a C++ complete program** that implements the queue using linked list, the following operations must be implement in the program **(Insert Elements, Delete Elements, Count Elements and Display Elements)?** **(20 Marks)**

```
#include<iostream>
#include<cstdlib>
using namespace std;
struct node {
   int info;
   struct node *next;
}*front,*rear;

void enqueue(int n);
void dequeue();
void display();
int count ();

int main() {
   int ch,n;
   rear=NULL;
   front=NULL;
   while(1)
   {
      cout<<"\n\n\tMENU";
      cout<<"\n##############################";
      cout<<"\n1. Insert\n2. Delete\n3. Display\n4. Exit\n";
      cout<<"\nEnter Your Choice: ";
      cin>>ch;
      switch(ch)
      {
         case 1:
         cout<<"\nEnter The Element: \n";
         cin>>n;
         enqueue(n);
         display();
         break;
         case 2:
         dequeue();
         break;
         case 3:
         display();
         break;
         case 4:
         count ();
         break;
         case 5:
         exit(0);
         break;
         default:
         cout<<"\nWrong Choice!!! Try Again.";
      }
   }
   return 0;
}
```

```
void enqueue(int n) {
   node *p;
   p=new node[sizeof(node)];
   p->info=n;
   p->next=NULL;
   if(rear==NULL||front==NULL)
   front=p;
   else
   rear->next=p;
   rear=p;
}

void dequeue() {
   node *p;
   int n;
   if(front==NULL||rear==NULL)
   {       cout<<"\nQueue Empty!!!";    }
   else
   {
      p=front;
      n=p->info;
      front=front->next;
      delete(p);
      cout<<"The Deleted Element = "<<n<<"\n";
      display();
   }
}

void display() {
   node *t;
   t=front;
   if(front==NULL||rear==NULL)
   {
      cout<<"\nQueue Empty!!!";
   }
   else
   {
     cout<<"\nQueue Elements are:\n";
     while(t!=NULL)
     {
        cout<<t->info<<"\n";
        t=t->next;
     }
   }
}
int count() {
   node *t;
   int c = 0;
   for (t = front ; t != NULL; t = t -> next)
      c++;
 return c;
}
```

Q8: Suppose we have **an array of 7 elements** (10, 20, 50, 30, 40, 70, 60), **Write a C++ function** that illustrate how to search about item 20, 30, 70 and 100 in the array A using **BINARY SEARCH TECHNIQUE?** **(20 Marks)**

BINARY SEARCH Algorithm

1. Input an array A of n elements and data to be searched.
2. LB = 0, UB = n-1; mid = int ((LB+UB)/2)
3. Repeat step 4 : 6 while (LB <= UB) and (A[mid] ! = data)
4. If (data < A[mid])
   (a) UB = mid–1
5. Else
   (a) LB = mid + 1
6. mid = int ((LB + UB)/2)
7. If (A[mid]== data)
   (a) Display “the data found at mid”
8. Else
   (a) Display “the data is not found”
9. Exit

```
#include<iostream>
using namespace std;

int binarySearch (int AR[], int N, int VAL);
int main()
{
	int AR[100],n,val,found;
	cout<<"Enter number of elements you want to insert ";
	cin>>n;
	cout<<"Enter element in ascending order\n";
	for(int i=0;i<n;i++)
	{
		cout<<"Enter element "<<i+1<<":";
		cin>>AR[i];
	}
	cout<<"\nEnter the number you want to search ";
	cin>>val;
	found= binarySearch (AR,n,val);
	if(found==1)
		cout<<"\nItem found";
	else
		cout<<"\nItem not found";

	return 0;
}

int binarySearch(int AR[], int N, int VAL)
{
	int Mid, Lbound=0,Ubound=N-1;

	while(Lbound<=Ubound)
	{
		Mid=(Lbound+Ubound)/2;
		if(VAL>AR[Mid])
			Lbound=Mid+1;
		else if(VAL<AR[Mid])
			Ubound=Mid-1;
		else
			return 1;
	}

	return 0;
}
```

Q9: Bubble sort algorithm is inefficient because it continues execution even after an array is sorted by performing unnecessary comparisons. Therefore, the numbers of comparisons in the best and worst cases are the same. **Modify the algorithm** *in such a fashion that it will not make the next pass when the array is already sorted?* **(20 Marks)**

Ans Q2:

Bubble sort continues execution even after an array is sorted. To prevent unnecessary comparisons we add a Boolean variable say switched and initialize it by True in the beginning. Along with the "for" loop, we add the condition (switched=true) and make it false inside the outer for loop. If a swapping is made then the value of switched is made true. Thus if no swapping is done in the first pass, no more comparisons will be done further and the program shall exit. ***The algorithm after modifying IS:-***

```
void bubble(int x[],int n)
{
int j,pass,hold;
bool switched=true;
for(pass=0;pass<n-1 && switched=true;pass++)
{
switched=false;
for(j=0;j<n-pass-1;j++)
{
switched=true;
hold=x[j];

x[j]=x[j+1];

x[j+1]=hold;
}
}
}
```

Good Luck,