* 1. Data Structure Quiz 1

1. What is algorithm?

(1) Give a definition of algorithm. (3%)

(2) What the criteria of algorithm. (3%)

(3) What is the ADT (Abstract Data Type)? (3%)

1. Please compare the time complexity and space complexity of "2D Array", " Transpose a Matrix" and " Fast Transpose Matrix". (6%)

|  |  |  |
| --- | --- | --- |
|  | space | time |
| 2D Array |  |  |
| Transpose a Matrix |  |  |
| Fast Transpose Matrix |  |  |

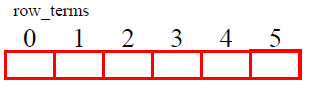
1. (1) Please refer to Table\_1 to complete Table\_2. Please use a fast transpose matrix way and must write down the result. (2%)

(2) Please use row\_term of Table\_2 to finish Table\_3. (3%)

Table\_1

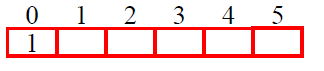
|  |  |  |  |
| --- | --- | --- | --- |
|  | row | col | value |
| a[0] | 6 | 6 | 8 |
| a[1] | 0 | 0 | 15 |
| a[2] | 0 | 1 | 22 |
| a[3] | 0 | 3 | -15 |
| a[4] | 1 | 5 | 11 |
| a[5] | 1 | 0 | 3 |
| a[6] | 2 | 2 | -6 |
| a[7] | 4 | 0 | 91 |
| a[8] | 5 | 2 | 28 |

Table\_2



Table\_3

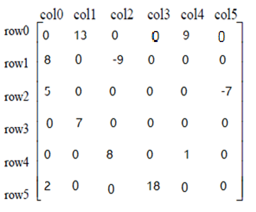
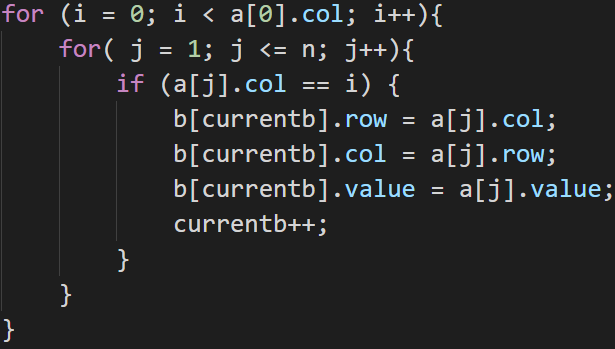
start\_pos



1. Sparse matrix and code are shown below. (n = a[0].value).

(1) Please use this matrix fill the Table\_1. Table 1 sorted by row from small to large. (3%)

(2) Please use code to transpose matrix and fill the Table\_2. (3%)

Table\_1 Table\_2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | row | col | value |  |  | row | col | value |
| a[0] | 6 | 6 | 11 |  | b[0] | 6 | 6 | 11 |
| a[1] |  |  |  |  | b[1] |  |  |  |
| a[2] |  |  |  |  | b[2] |  |  |  |
| a[3] |  |  |  |  | b[3] |  |  |  |
| a[4] |  |  |  |  | b[4] |  |  |  |
| a[5] |  |  |  |  | b[5] |  |  |  |
| a[6] |  |  |  |  | b[6] |  |  |  |
| a[7] |  |  |  |  | b[7] |  |  |  |
| a[8] |  |  |  |  | b[8] |  |  |  |
| a[9] |  |  |  |  | b[9] |  |  |  |
| a[10] |  |  |  |  | b[10] |  |  |  |
| a[11] |  |  |  |  | b[11] |  |  |  |

1. Give you a string like “([[]()])”, if ‘(‘ or ‘[‘ can match the ‘)’ or ‘]’ code print “Yes”. Otherwise, code print “No”. Please follow the rules and write down your code**.**  (8%)

|  |
| --- |
| #include<string.h>  #include<stdio.h>  char stack[10001];  char s[10001];  int main()  {  int n;  int top=1,len;  scanf("%s", &s);  len=strlen(s);  stack[top]=s[0];  top++;  for(int i=1;i<len;i++)  {  if(s[i] == '[' || s[i] == '(')  stack[top++]=s[i];  else  {  if(stack[top-1]=='[' && s[i]==']')  top--;  else if(stack[top-1]=='(' && s[i]==')')  **/\* your answer \*/ (a)**  else  stack[**/\* your answer \*/**]=s[i]; (b)  }  }  if(top==1)  printf("Yes\n");  else  printf("No\n");    return 0;  } |

1. Prefix, Infix, Postfix. (5%)
2. Write the postfix form of the following expression.

|  |
| --- |
| A + B \* C – D / E |

1. Write the prefix form of the following expression.

|  |
| --- |
| A + (B – C) / D – E / (F \* G) |

1. Write the infix form of the following expression.

|  |
| --- |
| A B C \* E 2 - / F \* + |

1. Write the infix form of the following expression.

|  |
| --- |
| / + A B \* - C D E |

1. Now I redefine the priority of the operator.

From high to low: “(“ , “)” higher than “+” , “-“ higher than “\*” , “/”.

Write the prefix form of the following expression.

|  |
| --- |
| A + B – C \* D / F \* G + H |

1. Time complexity sort. (6%)

O(n!) , O() , O() , O() , O(nn) , O(n*log*) , O(n*log*n) , O(1) , O(2n) , O() , O() , O() , O(n)

1. Write down the time complexity using big O. (5%)

(1) T(n) = T(n/2) + 1

(2)

(3)

(4) 2+

(5) +

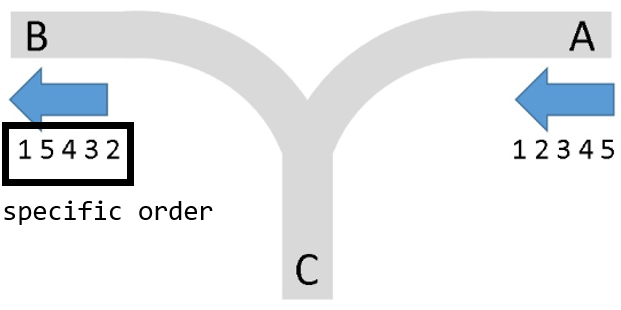
1. Please sum numbers from 1 to n, like 1+2+3+4+…+n. Please finish code by recursive function. (6%)

|  |
| --- |
| #include<stdio.h>  // recursive  int sum(int n)  {  if (n == 0)  return 0;  else  return sum(**/\* your answer \*/**) + n; (a)  }  int main()  {  int n;  printf("input sum number n\n");  scanf("%d", &n);  printf("Answer : %d" , sum(n));  return 0;  } |

1. Rail problem

Suppose **N** cars are entering from direction A. Its number is fixed to (1,2, 3, …N). Your task is to judge if these cars can leave to the direction B in a **specific order**.

For example, according to the above rules, this picture shows **N** = 5. If **specific order** = (1,2,3,4,5) and (1,5,4,3,2) are all feasible departure orders (true). If **specific order** = (5,4,1,2,3) is not feasible (false).



1. According to the following message to answer questions. Can the train leave to direction B in a specific order?

If the answer is true: Write “true”.

If the answer is false: Write “false”.

(a) **N** = 5, **specific order** = (1,4,2,5,3) (1%)

(b) **N** = 7, **specific order** = (1,6,2,5,4,3,7) (1%)

(c)  **N** = 9, **specific order** = (1,9,2,8,3,7,4,6,5) (1%)

1. N = 4. Write all specific orders that can leave from direction B. (3%)
2. The node\_x is a user-defined structure and list\_y is a pointer type that points to a linked list.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  typedef struct node{  char chData;  struct node \*link;  } node\_x, \*list\_y;    void hard (list\_y \*ptr, char \*s){  if(\*s){  node\_x \*p = malloc (sizeof (node\_x));  p->chData = \*s;  if((int)s[0]%5 == 0)  hard(&p->link, s+5);  else  hard(&p->link, s+((int)s[0]%5));  printf("%c", p->chData);  \*ptr = &p+1;  }  else  \*ptr = NULL;  }  int main( ){  list\_y p = NULL;  hard (&p, "structure");  } |

(1) Write down the output. (3%)

(2) Time complexity? (5%)

1. Adding polynomials.
2. Please use linked list to implement this polynomial. (1%)

a = 11+9+6-108-17

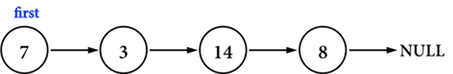
(2) Please finish the function below. (6%)

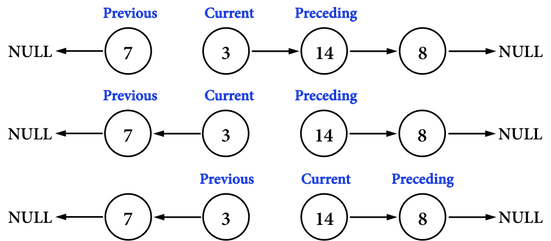
|  |
| --- |
| typedef struct polyNode \*polyPointer;  typedef struct polyNode  {  int coef;  int expon;  polyPointer link;  };  polyPointer a , b;  polyPointer padd(polyPointer a , polyPointer b)  {//return a polynomial = a+b  polyPointer c , rear , temp;  int sum;  malloc (rear , sizeof(\*rear));  c = rear;  while( a && b){  switch(compare(a->expon , b->expon))  {  case -1: //a -> expon < b->expon  attach(b->coef, b->expon, &rear);  b = b->link;  break;  case 0: //a -> expon = b->expon  **/\* your answers \*/** (a)  case 1: //a-> expon > b->expon  **/\* your answers \*/ (b)**  }  }  /\* copy rest of list \*/  for( ; a ; a = a->link)  attach(a->coef , a->expon , &rear);  for( ; b ; b = b->link)  attach(b->coef , b->expon , &rear);  rear -> link = NULL;  /\*delete extra initial node\*/  temp = c;  c = c->link;  free(temp);  return c;  } |

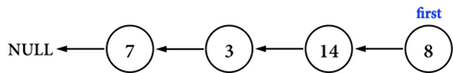
1. I want you to create stack and queue by linked lists. Stack should have push and pop functions. Queue should have enqueue and dequeue functions. Please finish the program below. (8%)

|  |
| --- |
| #include <stdio.h>  void stack\_push(int i , element item)  {  stackPointer temp;  malloc (temp , sizeof(\*temp));  temp->data = item;  temp->link = top[i];  top[i] = temp;  }  element stack\_pop(int i)  {  **/\* your answers \*/ (a)**  free(temp);  return item;  }  void queue\_enqueue(int i , element item)  {  queuePointer temp;  malloc (temp , sizeof(\*temp));  temp->data = item;  temp->link = NULL;  if (front[i])  rear[i]->link = temp;  else  front[i] = temp;  rear[i] = temp;  }  element queue\_ dequeue (int i)  {  **/\* your answers \*/ (b)**  free(temp);  return item;  }  int main()  {  int i ;  element item;  stack\_push(i , item);  stack\_pop(i);  queue\_ enqueue (i , item);  queue\_ dequeue (i);  } |

1. Please use C program to finish the code. (8%)



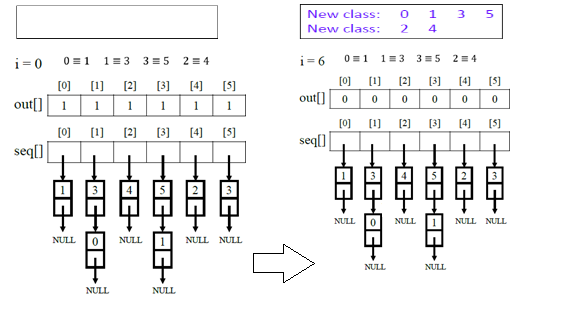




|  |
| --- |
| void LinkedListReverse()  {  if (first == 0 || first->next == 0) {  // list is empty or list has only one node  return;  }  ListNode \*previous = 0, \*current = first, \*preceding = first->next;  while (preceding != 0) {  **/\* your answer \*/** // turn around current->next (a)  **/\* your answer \*/** // put back previous (b)  **/\* your answer \*/** // put back current (c)  **/\* your answer \*/** // put back preceding (d)  }  current->next = previous;  first = current;  } |

1. Please refer to Figure 1 and Figure 2 to complete the program below. (7%)

Figure 1 Figure 2



|  |
| --- |
| for (i = 0 ; i < n ; i++)  {  if (out[i])  {  printf(“\n new class : %5d” , i);  out[i] = FALSE;  x = seq[i];  top = NULL;  for(;;)  {  while(x)  {  j = x->data;  if (out[j])  {  printf(“%d” , j);  **/\* your answers \*/ (a)**  }  else  {  x = x->link;  }  }  if (!top)  break;  x = seq[top->data];  top = top->link;  }  }  } |