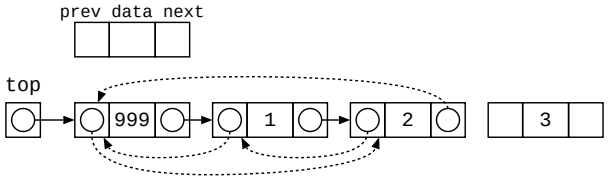
情報構造論 (斉藤)　後期中間試験　　番号：
4EI　2020/12/7　(1/3)　　氏名：

以下の 5 つの設問の中から、4 つを選んで解答せよ。

1　双方向循環リスト Deque(25)

以下のプログラム, 図に示すような、双方向循環リストのプログラムを作りたい。
Deque の先頭からデータを取り出す shift(), 末尾にデータを追加する push() を完成させたい。
図の中に、データを取り出した後のポインタの繋がり方を記載 (8) し、プログラム中の (A)-(F) をうめよ (17)。




```
struct Deque* dqcons( struct Deque* p , int x , struct Deque* n ) {
   struct Deque* a ;
   a = (struct Deque*)malloc( sizeof( struct Deque ) ) ;
   if ( a != NULL ) {
      a->prev = p ;                | struct Deque {
      a->data = x ;                |    int data ;
      a->next = n ;                |    struct Deque* prev ;
   }                               |    struct Deque* next ;
   return a ;                      | } ;
}

int shift( struct Deque* top ) {   | int main() {
   int ans = top->next->data ;     |    struct Deque* top
   struct Deque* d = top->next ;   |      = dqcons( NULL, 999,
                                   |           dqcons( NULL, 1,
   d->next->        = top ;        |              dqcons( NULL, 2, NULL ) ) ) ;
            ~~~~~~~~(A)3           |    top->next->prev = top ;
   top->next =                     |    top->next->next->prev = top->next ;
              ~~~~~~~~~~~(B)3      |    top->prev = top->next->next ;
                                   |    top->next->next->next = top ;
   ~~~~~~~~~~~~~~~(C)3             |    push( top , 3 ) ;
   return ans ;                    |    printf( "%d\n" , shift( top ) ) ;
}                                  |    return 0 ;       // 1 を取出す
                                   | }
void push(                    top , int x ) {
          ~~~~~~~~~~~~~~~~~~(D)2
   struct Deque* n = dqcons(              , x , top ) ;
                           ~~~~~~~~~~~(E)3
                        = n ;
   ~~~~~~~~~~~~~~~~~~~(F)3
   top->prev = n ;
}
```

2　説明問題 (25)

1. 前設問で、双方向リストの先頭に 999 といった、本来保存対象でないデータを置く手法の名前と、その手法を用いる理由を説明せよ。

2. int a[] = { 53, 11, 86, 10, 22, 65, 92 } ; のようなデータ順で保存する 2 分ヒープについて特徴を説明し、データ件数が N の場合の検索時間のオーダについて説明せよ。(13)

3　正規表現

1. 正規表現 [0-9+[A-Z]*が表現する文字列の集合の要素となるものはどれか。要素となるものに◯、ならないものに×を記入せよ。

   (1) 45678 ______ (2) 99ABC* ______ (3) 987+BA ______ (4) ZYXWV ______

2. 次の BNF の規則から、生成することができる式に◯、ならないものに×を記入せよ。

```
<exp> ::= <var>                    <...> , ::= , | , ;
        | ( <exp> + <exp> )           は BNF 記法の記号とする
        |   <exp> * <exp>          ::= は定義, | は選択
        ;                          ; は BNF 記法の終端とする
<var> ::= A | B | C | D ;
```

   (1) A+(B+C)*D _____ (2) (A+B)+(C+D) _____ (3) (A+B)*(C+D) _____ (4) (A*B)+(C*D) _____

4　構文木と逆ポーランド記法 (25)

以下のようなプログラムで、main() に示すように、1+2*3 に相当する構文木を生成する。
このような式を、逆ポーランド記法で式を出力する関数 print_rpn() を作成せよ。

```
struct Tree {              | void main() { // 1 + 2 * 3
   int data ;              |    struct Tree* top
   struct Tree* left ;     |      = t_op( '+' , t_int(1) ,
   struct Tree* right ;    |              t_op( '*' , t_int(2) , t_int(3) ) ) ;
} ;                        |    print_rpn( top ) ;
                           | }

struct Tree* t_op( int op , struct Tree* L , struct Tree* R ) {
   struct Tree* ans ;
   ans = (struct Tree*)malloc( sizeof( struct Tree ) ) ;
   if ( ans != NULL ) {
      ans->data = op ;
      ans->left = L ;      | struct Tree* t_int( int x ) {
      ans->right = R ;     |    return t_op( x , NULL , NULL ) ;
   }                       | }
   return ans ;
}

void print_rpn( ~~~~~~~~~~~~~~~~~~~~~~~~ ) {
```

5　2分探索木の応用

```
struct PNTree {                    | 電話番号と名前のデータを
   int            phone ;          | 電話番号をキーとする２分探索木
   char           name[ 20 ] ;     | に保存するプログラムを作成した。
   struct PNTree* left ;           | 電話番号が若いものが左の枝に
   struct PNTree* right ;          | くるように、穴埋めすること。
} ;                                | また、電話番号で検索する関数、
                                   | 名前で検索する関数を完成させよ。

void add_phone_name( int ph , char nm[] ) {
   struct PNTree** tail = &top ;
   while( *tail != NULL ) {

      if ( (*tail)->data             )
                        ~~~~~~~~~~~~(A)2
         return ;
      else if (                          )
               ~~~~~~~~~~~~~~~~~~~~~~~~~~(B)3
         tail = &( (*tail)->left ) ;
      else
                                    ;
   }     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~(C)3
   *tail =
          ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~(D)3
   if ( *tail != NULL ) {
      (*tail)->phone = ph ;
      strcpy( (*tail)->name , nm ) ;
      (*tail)->left = (*tail)->right = NULL ;
   }
}

int find_by_phone( struct PNTree* p ,
                   int ph ) {
   while(                    ) {
          ~~~~~~~~~~~~~~~~~~(E)3
      if ( p->phone == ph )
         return 1 ;
      else if ( p->phone > ph )

         ~~~~~~~~~~~~~~~~~~~(F)3
      else
   }     ~~~~~~~~~~~~~~~~~~~(G)3
   return 0 ;
}

struct PNTree* top = NULL ;

void main() {
   add_phone_name( 272727 , "nobody" ) ;
   add_phone_name( 210110 , "police" ) ;
   add_phone_name( 621111 , "fnct" ) ;

   if ( find_by_phone( top , 210110 ) ) printf( "Find 210110\n" ) ;
   if ( find_by_name( top , "fnct" ) ) printf( "Find fnct\n" ) ;
}
```

```
int find_by_name( struct PNTree* p ,
                  char nm[] ) {
   if ( p == NULL ) {
      return 0 ;
   } else {
      if ( strcmp( p->name,nm) == 0 )
         return 1 ;
      else if ( find_by_name(             , nm ) )
                              ~~~~~~~~~~~(H)2
         return 1 ;
      else if (                                    )
               ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~(I)3
         return 1 ;
      else
         return 0 ;
   }
}
```