

## DS Lab 4 Solutions

### 4.1.1

```
1  #include<stdio.h>
2  #include<string.h>
3  #include<stdlib.h>
4  #include<ctype.h>
5  #include<stdbool.h>
6  #define N 20
7  char s[20];
8  int top = -1;
9  int isEmpty()
10 v {
11     —> if(top < 0)
12     —> return 1;
13     —> else
14     —> return 0;
15 }
16 void push(char x)
17 v {
18     —> if(top == N-1)
19 v —> {
20     —> —> printf("Stack is overflow.\n");
21     —> }
22     —> else
23 v —> {
24     —> —> top = top + 1;
25     —> —> s[top] = x;
26     —> }
27 }
```

```

28     char·pop()
29     v {
30         —>|if(top<0)
31     v —>|{
32         —>|—>|printf("Stack·is·underflow·:·unbalanced·parenthesis\n");
33         —>|—>|exit(0);
34         —>|}
35         —>|else
36         —>|return·s[top--];
37     }
38     int·priority(char·x)
39     v {
40         —>|if(x=='(')
41         —>|return·0;
42         —>|if(x=='+' || x=='-')
43         —>|return·1;
44         —>|if(x=='*' || x=='/' || x=='%')
45         —>|return·2;
46     }
47     bool·isalphanum(char·a)
48     v {
49         —>|if((a>=65&&a<=90) || (a>=97&&a<=122))
50         —>|return·true;
51         —>|return·false;
52     }

```

```

53 void convertInfix(char *e)
54 {
55     int x;
56     int k = 0;
57     char *p = (char *) malloc(sizeof(char) * strlen(e));
58     while(*e != '\0')
59     {
60         if(isalphanum(*e))
61         {
62             p[k++] = *e;
63         }
64     }
65     else if(*e == '(')
66     {
67         push(*e);
68     }
69     else if(*e == ')')
70     {
71         while(!isEmpty() && (x = pop()) != '(')
72             p[k++] = x;
73     }
74     else if(*e == '+' || *e == '-' || *e == '*' || *e == '/' || *e == '%')
75     {
76         while(priority(s[top]) >= priority(*e))

```

```

77             p[k++] = pop();
78             push(*e);
79         }
80     }
81     else {
82         printf("Invalid symbols in infix expression. Only alphanumeric characters, parenthesis and valid arithmetic operators are allowed.\n");
83         exit(0);
84     }
85     e++;
86 }
87 }
88 while(top != -1)
89 {
90     x = pop();
91     if(x == '(')
92     {
93         printf("Invalid infix expression :: unbalanced parenthesis.\n");
94         exit(0);
95     }
96     p[k++] = x;
97 }
98 p[k++] = '\0';
99 printf("Postfix expression : %s\n", p);
100 }
101 }

```

#### 4.1.2

```

1  #include<stdio.h>
2  v int main(){
3      —>int i=0,a[100],choice,x=0;
4      —>printf("Enter the choice for stack operation: \n");
5      —>printf("press 1 for push\npress 2 for pop\npress 3 for display\npress 4
      for exit\n");
6      —>scanf("%d",&choice);
7  v —>while(choice!=4){
8  v —>—>switch(choice){
9      —>—>—>case 1:
10     —>—>—>—>printf("Enter item to insert: ");
11     —>—>—>—>scanf("%d",&a[i]);
12     —>—>—>—>printf("Item inserted = %d\n",a[i]);
13     —>—>—>—>i++;
14     —>—>—>—>break;
15     —>—>—>case 2:
16  v —>—>—>—>—>if(i==0){
17     —>—>—>—>—>printf("Stack underflow\n");
18     —>—>—>—>—>}
19  v —>—>—>—>—>else{
20     —>—>—>—>—>printf("Item deleted = %d\n",a[i-1]);
21     —>—>—>—>—>a[i-1]=0;
22     —>—>—>—>—>i--;
23     —>—>—>—>—>}
24     —>—>—>—>—>break;
25     —>—>—>case 3:
26  v —>—>—>—>—>if(i==0){
27     —>—>—>—>—>printf("No more data\n");

```

```

27     printf("No more data\n");
28     break;
29 }
30 printf("Stack are\n");
31 x=i;
32 v while(i>0){
33     printf("%d ",a[i-1]);
34     i--;
35 }
36 printf("\n");
37 i=x;
38 break;
39 }
40 printf("Enter the choice for stack operation: ");
41 scanf("%d",&choice);
42 }
43 }
44
45

```

#### 4.1.3

```

1  #include<stdio.h>
2
3  v int read(int *a,int m,int n){
4      —>printf("Enter %d elements : ",m*n);
5  v —>for(int i=0;i<m;i++){
6  v —>—>for(int j=0;j<n;j++){
7      —>—>—>scanf("%d",&(a+i+j));
8      —>—>}
9      —>}
10 }
11 v int display(int *a,int m,int n){
12 v —>for(int i=0;i<m;i++){
13 v —>—>for(int j=0;j<n;j++){
14     —>—>—>printf("%d ",*(a+i+j));
15     —>—>}
16     —>—>printf("\n");
17     —>}
18 }
19
20 v int multiplicationOfTwoMatrices(int *a,int *b,int m,int n,int q){
21     ...printf("The Multiplication Matrix is\n");
22 v —>for(int i=0;i<m;i++){
23 v —>—>for(int j=0;j<q;j++){
24     —>—>—>int sum=0;
25 v —>—>—>for(int k=0;k<n;k++){
26     —>—>—>sum+=((*(a+i+k))*(*(b+k+j)));
27     —>—>—>}
28     —>—>—>printf("%d ",sum);
29     —>—>}
30     —>—>printf("\n");
31     —>}
32 }

```

#### 4.2.1

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4  v typedef struct Stack{
5      —>int top;
6      —>unsigned capacity;
7      —>char* array;
8  }Stack;
9
10 v Stack* createStack(unsigned capacity){
11     —>Stack* stack=(Stack*)malloc(sizeof(Stack));
12     —>stack->capacity=capacity;
13     —>stack->top=-1;
14     —>stack->array=(char*)malloc(stack->capacity*sizeof(char));
15     —>return stack;
16 }
17 v int isFull(Stack* stack){
18     —>return stack->top==stack->capacity-1;
19 }
20 v int isEmpty(Stack* stack){
21     —>return stack->top==-1;
22 }
23 v void push(Stack* stack, char item){
24     —>if(isFull(stack))
25     —>—>return;
26     —>stack->array[++stack->top]=item;
27 }
28 v void pop(Stack* stack){

```

```

28  v char·pop(Stack*·stack){
29      —> if(isEmpty(stack))
30      —> —> return·-1;
31      —> return·stack->array[stack->top--];
32      }
33  v int·isMatchingPair(char·character1,char·character2){
34  v —> if(character1=='('&&character2==')'){
35      —> —> return·-1;
36      —> —>
37      —> }
38      —> else
39      —> —> return·0;
40      }
41
42  v int·areParenthesisBalanced(char·exp[]){
43      —> int·i=0;
44      —> Stack*·stack=createStack(strlen(exp));
45  v —> while(exp[i]){
46      —> —> if(exp[i]=='(')
47      —> —> —> push(stack,exp[i]);
48  v —> —> else·if(exp[i]==')'){
49      —> —> —> if(isEmpty(stack))
50      —> —> —> —> return·0;
51  v —> —> —> else·if(!isMatchingPair(pop(stack),exp[i])){
52      —> —> —> —> return·0;
53      —> —> —> }
54      —> —> }

```

```

55      —> —> i++;
56      —> }
57      —> if(isEmpty(stack))
58      —> —> return·1;
59      —> else
60      —> —> return·0;
61      }
62  v int·main(){
63      —> char·exp[100];
64      —> printf("Enter·a·bracket·sequence:·");
65      —> scanf("%s",exp);
66      —> if(areParenthesisBalanced(exp))
67      —> —> printf("Balanced·brackets\n");
68      —> else
69      —> —> printf("Not·balanced·brackets\n");
70      }

```

#### 4.2.2



```

1  #include<stdio.h>
2  #define SIZE 100
3
4
5  v void printNGE(int arr[],int n){
6      —>int next,i,j;
7      —>printf("Input: ");
8  v —>for(i=0;i<n;i++){
9      —>—>next=-1;
10 v —>—>for(j=i+1;j<n;j++){
11 v —>—>—>if(arr[i]<arr[j]){
12     —>—>—>next=arr[j];
13     —>—>—>break;
14     —>—>—>}
15     —>—>}
16     —>—>printf("%d ",arr[i]);
17     —>—>
18     —>}
19     —>printf("\n");
20     —>printf("Output: ");
21 v —>for(i=0;i<n;i++){
22     —>—>next=-1;
23 v —>—>for(j=i+1;j<n;j++){
24 v —>—>—>if(arr[i]<arr[j]){
25     —>—>—>next=arr[j];
26     —>—>—>break;
27     —>—>—>}
28     —>—>}

```

```

29     —>—>printf("%d ",next);
30     —>}
31     —>printf("\n");
32 }
33 v void input(){
34     —>printf("Input: ");
35 }
36 v int main(){
37     —>int arr[SIZE],n,i;
38     —>printf("Enter n: ");
39     —>scanf("%d",&n);
40     —>printf("Enter n elements: ");
41 v —>for(i=0;i<n;i++){
42     —>—>scanf("%d",&arr[i]);
43     —>}
44     —>printNGE(arr,n);
45 }

```

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  v void findNextGreaterElement(int arr[],int n){
4      —>int next[n],i,j;
5  v —>for(i=0;i<n;i++){
6      —>—>next[i]=-1;
7      —>}
8  v —>for(i=0;i<n;i++){
9  v —>—>for(j=i+1;j<n;j++){
10 v —>—>—>if(arr[j]>arr[i]){
11 —>—>—>—>next[i]=arr[j];
12 —>—>—>—>break;
13 —>—>—>}
14 —>—>}
15 —>}
16 v —>for(i=0;i<n;i++){
17 —>—>printf("%d ",next[i]);
18 —>}
19 }
20 v int main(){
21 —>int n,i;
22 —>printf("Enter n: ");
23 —>scanf("%d",&n);
24 —>int arr[n];
25 —>printf("Enter n elements: ");
26 v —>for(i=0;i<n;i++){
27 —>—>scanf("%d",&arr[i]);
28 —>}

```

```

29 —>printf("Input: ");
30 v —>for(i=0;i<n;i++){
31 —>—>printf("%d ",arr[i]);
32 —>}
33 —>printf("\nOutput: ");
34 —>findNextGreaterElement(arr,n);
35 —>printf("\n");
36 }

```