ASSIGNMENT 2

G.SINDHU

24011A510

1.

```
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int top=-1;
char  infix[100];
char postfix[100];
int precedence(char element)
{
   switch(element)
   {
      case '+':
      case '-':
      return 1;
      case '*':
      case '/':
      return 1;
      case '@':
      return 2;
      case '^':
      return 3;
      default:
      return 0;
   }
}
int isoperator(char a)
{
   return a=='+'||a=='-'||a=='/'||a=='*'||a=='@'||a=='^';
}
void push(char element)
{
   top=top+1;
   infix[top]=element;
}
char pop()
{
   char h;
  if (top==-1)
  return -1;
  else
```

```
  h=infix[top];
  top=top-1;
  return h;
}
int isempty()
{
   if(top==-1)
   {
      return 0;
   }
   else
   {
      return 1;
   }
}
int peek()
{
   return infix[top];
}
int main() {
   int i,j=0;
   char h[100];
   char k;
 printf("\nenter the infix expression:");
 scanf("%s",&h);
 for(i=0;i<strlen(h);i++)
 {
   if(isalnum(h[i]))
   {
      postfix[j]=h[i];
      j++;
   }
   else if(h[i]=='(')
   {
       push(h[i]);
   }
  else if(h[i]==')')
  {
     while(isempty())
     {
        k=pop();
        if(k=='(')
        {
        break;
        }
        else
        {
```

```
        postfix[j]=k;
        j++;
        }
     }
  }
  else if(isoperator(h[i]))
  {
     while(isempty() && precedence(infix[top])>=precedence(h[i]))
     {
        postfix[j]=pop();
        j++;
     }
     push(h[i]);
  }
 }
while(isempty())
{
   postfix[j]=pop();
   j++;
}
postfix[j]='\0';
printf("\nThe postfix expression is:%s",postfix);
}
```

2.

```
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

int top=-1;
char stack[100];

int isnotempty()
{
   if(top==-1)
   return 0;
   else
   return 1;
}
void push(char element)
{
   top=top+1;
   stack[top]=element;
}
char pop()
{
```

```
    char k;
    if(top==-1)
    return '\0';
    else
    {
       k=stack[top];
       top=top-1;
       return k;
    }
}
int main() {
   char k[100],z;
   int i;
   printf("\nenter comment:");
   scanf("%99s",&k);
   for(i=0;i<strlen(k);i++)
   {
     if(k[i]=='}')
     {
        while(isnotempty())
        {
           z=pop();
           if(z=='{')
           break;
        }
     }
     else
     {
        push(k[i]);
     }
   }
   if(isnotempty())
   printf("\nThe comment is not a valid nesting");
   else
   printf("\nThe comment follows valid nesting");
    return 0;
}
3.
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int top=-1;
int stack[100];
int minstack[100];
int minimum=-1;
```

```
void push(int element)
{
  if(top==-1)
  {
     top=top+1;
     minimum=minimum+1;
     stack[top]=element;
     minstack[top]=element;
  }
  else
  {
     top=top+1;
     stack[top]=element;
     if(minstack[minimum]>element)
     minstack[++minimum]=element;
  }
}
void pop()
{
  if (top==-1)
  printf("\nTHE stack is empty");
  else
  {
     if(stack[top]==minstack[minimum])
     {
        --minimum;
        top=top-1;
     }
     else
     top=top-1;
  }
}
int findmin()
{
  return minstack[minimum];
}
int main() {
  push(30);
  pop();
  push(70);
  pop();
  push(90);
  push(2);
  push(-34);
  push(6);
  pop();
```

```
  printf("\nThe minimum of the stack is:%d",findmin());
  push(90);
  push(-190);
  push(1);
  pop();
  pop();
  push(1);
  printf("\nThe minimum of the stack is:%d",findmin());
  return 0;
}
4.
// Online C compiler to run C program online
#include <stdio.h>
#include<stdlib.h>
//t front=-1,rear=-1;
int tmp=0,n=0;
struct student
{
   int data;
   struct student*next;
}*front=NULL,*rear=NULL,*newnode=NULL;
void enqueue(int size,int element)
{
   if(tmp<size)
   {
   newnode=((struct student*)malloc(sizeof(struct student)));
   newnode->data=element;
   newnode->next=NULL;
   tmp++;
   n++;
  if(rear)
  {
    rear->next=newnode;
    rear=newnode;
  }
  else
  {
     rear=front=newnode;
  }
  }
  else
  {
     printf("\n Queue is full");
  }
}
void dequeue()
{
```

```
   if(front&& front==rear)
   {
      newnode=front;
      front=rear=NULL;
      free(newnode);
      n--;
   }
   else if(front)
   {
      newnode=front;
      front=front->next;
      free(newnode);
      n--;
   }
   else if(!front)
   {
      printf("\nqueue is empty");
   }
}
void isempty()
{
   if(rear)
   {
      printf("\n QUEUE is not empty");
   }
   else
   {
      printf("\n Queue is not empty");
   }
}
int getsize()
{
   return n;
}
int peek()
{
   return front->data;
}
int main() {
   int size;
   printf("\nenter the size for the linear queue;");
   scanf("%d",&size);
   enqueue(size,4);
    enqueue(size,5);
    enqueue(size,9);
    dequeue();
     dequeue();
```

```
    isempty();
    enqueue(size,9);
    enqueue(size,9);
    enqueue(size,9);
    enqueue(size,9);
    enqueue(size,9);
      printf("\nGet the first element inserted:%d",peek());
      printf("\n THE size of queuee is:%d",getsize());

  return 0;
}
```

b)

```
// Online C compiler to run C program online
#include <stdio.h>
#include<stdlib.h>
int front=-1,rear=-1;
int a[100];
void enqueue(int size,int element)
{
  if(rear==size-1)
  printf("\n overflow");
  else if(rear==-1)
  {
    rear=rear+1;
    front=front+1;
    a[rear]=element;
  }
  else
  {
    rear=rear+1;
    a[rear]=element;
  }
}
void dequeue()
{
  if(front==-1)
  printf("Underflow");
  else if(front==rear)
  {
    front=rear=-1;
  }
  else
  {
    front=front+1;
  }
}
void isempty()
```

```
{
   if(front==-1)
   {
      printf("QUEUE IS EMPTY");
   }
   else
   {
      printf("QUEUE IS not EMPTy");
   }
}
int peek()
{
   return a[front];
}
int getsize()
{
   if(front)
   {
   return (rear-front)+1;
   }
   else
   return 0;
}
int main() {
   int size;
  printf("\nenter the size of the queue");
  scanf("%d",&size);
   enqueue(size,4);
    enqueue(size,5);
    enqueue(size,9);
    dequeue();
     dequeue();
     isempty();
     enqueue(size,9);
     enqueue(size,9);
     enqueue(size,9);
     enqueue(size,9);
     enqueue(size,9);
       printf("\nGet the first element inserted:%d",peek());
       printf("\n THE size of queuee is:%d",getsize());
   return 0;
}
```

5.

```
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
```

```
#include <string.h>
#define max 10
int a[max];
int front=-1;
int rear=-1;
void enquefront(int element)
{
   if((front==0&&rear==max-1)||(front==rear+1))
   {
      printf("\nQueue is full");
   }
   else if(front==-1&&rear==-1)
   {
      front=rear=0;
      a[front]=element;
   }
   else if(front==0)
   {
      front=max-1;
      a[front]=element;
   }
   else
   {
      front--;
      a[front]=element;
   }
}
void enquerear(int element)
{
   if((front==0&&rear==max-1)||(front==rear+1))
   {
      printf("\nQueue is full");
   }
   else if(front==-1&&rear==-1)
   {
      front=rear=0;
      a[rear]=element;
   }
   else if(rear==max-1)
   {
      rear=0;
      a[rear]=element;
   }
   else
   {
      rear++;
      a[rear]=element;
```

```
    }
}
void dequefront()
{
    if(front==-1&&rear==-1)
    {
        printf("\nthe queue is empty");
    }
    else if(front==rear)
    {
        front=rear=-1;
    }
    else if(front==max-1)
    {
        front=0;
    }
    else
    {
        front++;
    }
}
void dequerear()
{
    if(front==-1&&rear==-1)
    {
        printf("\nthe queue is empty");
    }
    else if(front==rear)
    {
        front=rear=-1;
    }
    else if(rear==0)
    {
        rear=max-1;
    }
    else
    {
        rear--;
    }
}
void displayfront()
{
 int i=front;
 while(i!=rear)
 {
     printf("\n%d",a[i]);
     i=(i+1)%max;
```

```
 }
 printf("\n%d",a[rear]);
}
int main() {
    enquerear(20);
    enquerear(60);
    enquefront(90);
    enquerear(2);
    enquefront(20);
    dequerear();
    dequefront();
    printf("\ndisplaying with front:");
    displayfront();
   enquefront(20);
   printf("\ndisplaying with front:");
    displayfront();
}
```